PROJECTE FI DE CARRERA

TÍTOL: **Desenvolupament d'una interfase sèrie de comunicacions per DSP**

AUTOR: Genís Caminal Villodres

TITULACIÓ: Enginyeria Tècnica Telecomunicacions

DIRECTOR: Sanchez Lopez Sergio

DEPARTAMENT: Departament d'arquitectura de computadors

CODIRECTOR: Blanqué Molina Balduí

CODIRECTOR: Gomila Gonzalez Marc

DATA: 4/8/2015

TÍTOL: **Desenvolupament d'una interfase sèrie de comunicacions per DSP**

COGNOMS: Caminal Villodres **NOM:** Genís

TITULACIÓ: Enginyeria Tècnica Telecomunicacions

ESPECIALITAT: Sistemes Electrònics **PLA: 95**

DIRECTOR: Sanchez Lopez, Sergio

DEPARTAMENT: Arquitectura de computadors

QUALIFICACIÓ DEL PFC

TRIBUNAL

PRESIDENT	SECRETARI	VOCAL
Simo Mezquita, Ester	Ponsa Asensio, Pedro	Arno Macia, Elisabet

DATA DE LECTURA: 24/09/2015

Aquest Projecte té en compte aspectes mediambientals: □ Sí ☑ No

PROJECTE FI DE CARRERA

RESUM (màxim 50 línies)

El present projecte final de carrera, tracta sobre la confecció i programació d'una plataforma amb un microcontrolador que es comunica via sèrie RS232 amb una DSP de control, o que realitza la funció d'inversor, connectada a un motor elèctric, i a més representa les dades mitjançant un LCD-TFT tàctil. Aquesta plataforma està formada per un arduino on es connecten dos mòduls (un per a realitzar la comunicació RS232 i l'altre el LCD tàctil per visualitzar les dades.

Aquesta memòria es pot dividir en tres grans apartats, en el primer apartat, s'introdueix i es resumeix l'estat dels inversors a la venda actualment per fabricants mundials destacats i s'exposen les diferències i novetats que incorpora aquesta nova plataforma amb arduino. EL segon apartat, que és el més gran, es detalla tot el disseny i el seu funcionament, tant a nivell hardware com software i l'apartat final, es demostra la connectivitat amb la DSP, s'exposen els problemes i les solucions aportades i les millores que es podrien arribar a realitzar.

Paraules clau (màxim 10):

Arduino	LCD-TFT	TouchScreen	RS-232
Comunicació	DSP	Open Source	Inversor
Panell Operari	Buffer Size		

SUMARI

1.Introducció.....6

1.1.Justificació del projecte.....6

1.2.Objectius a consolidar.....8

1.3.Estructura del treball.....9

2.Estat actual dels inversors.....10

2.1.Control·ladors de motor actuals disponibles, inversors.....10

2.2.Tendències de futur.....13

3.Descripció de la plataforma.....15

3.1.Diagrama de blocs del sistema.....15

3.2.Introducció als elements 16

4.Descripció del funcionament i programació del mòdul RS-232.....21

4.1.Comunicació RS-232 amb la DSP.....21

4.2.Programació del mòdul i primeres proves mitjançant MatLab.....27

5.Descripció del funcionament i programació del mòdul LCD-TFT tàctil.....36

5.1.Descripció driver TFT i de la touchscreen.....37

5.2.Funcions per detectar pulsacions a la touchscreen.....40

5.3.Funcions per dibuixar a la pantalla.....44

6.Demostració i explicació del programa complert.....56

7.Problemàtica i solucions aplicades.....66

8.Conclusions.....69

10.Bibliografía.....71

1.Introducció

1.1.Justificació del projecte

En el context actual, els inversors que alimenten i controlen motors elèctrics en corrent altern, són dispositius molt pesants, degut a tota la electrònica de potència que incorporen i normalment el dispositiu de control i maniobrabilitat es troba incorporat al mateix dispositiu formant tot un sistema o grup conjunt, on avui dia es pot connectar a un PC, aquí no es tindrà en compte ja que no és l'objectiu a consolidar, degut a que es necessita un dispositiu electrònic molt ràpid com són les DSP's; un aspecte nou i que es comença a veure en algunes marques, és la independència de la unitat de maniobrabilitat o control de l'usuari o operari. A la figura 1, es mostra l'estat actual de dos inversors comercials actualment a la venda.

Figura 1: A l'esquerra tenim un inversor marca Parker model A650 i a la dreta tenim un altre de la marca allen bradely powerflex70

Gràcies a l'aparició en els últims anys de noves tecnologies o grans millores tècniques, com per exemple: comunicació wifi, bluetooth, LCD's amb tecnologia TFT i pantalles tàctils, smartphones, etc. , es pot millorar molt el conjunt actual inversor-control d'usuari o operari i per tant amb aquest projecte, es començarà a aplicar aquest concepte utilitzant una plataforma amb microcontrolador idenpendent.

En el cas a aplicar en aquest projecte, tenim un motor elèctric, amb diferents sensors, controlat per un dispositiu DSC-DSP, amb un control molt complicat des del punt de vista de l'usuari no tècnic, ja que es realitza mitjançant un PC per llegir i escriure variables, realitzar gràfiques, etc., per tant, com a millora a la maniobrabilitat del dispositiu i com es comenta al paràgraf anterior, utilitzarem com a plataforma un microcontrolador independent, on el control i visualització de les dades del dispositiu, es realitzarà de forma gràfica amb una pantalla LCD-TFT tàctil i on el disseny del software estarà molt orientat a la interacció amb l'usuari de forma senzilla e intuïtiva.

La comunicació entre la plataforma i la DSC-DSP es realitza mitjançant RS232 i el disseny d'un protocol de comunicació propi, ja que encara no existeixen eines lliures per tal de comunicar les dues plataformes i que interactuïn conjuntament, a més d'aquesta manera es prepara el primer pas per a poder millorar aquest projecte a altres formes de comunicació més modernes. A la figura 2, es mostra el conjunt DSP + motor elèctric + part de la electrònica de potència del laboratori d'enginyeria elèctrica a la EPSEVG on es realitzaran les proves.

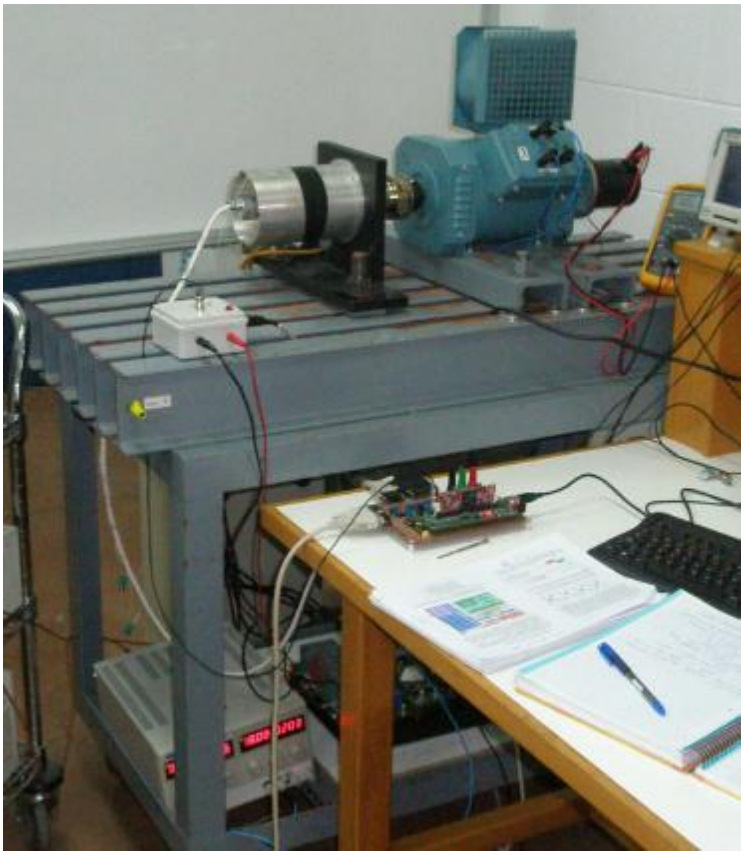


Figura 2: Motor elèctric + DSP de texas instruments

1.2.Objectius a consolidar

Com s'ha comentat a l'apartat anterior, l'objectiu vist de forma general es comunicar una plataforma amb microcontrolador amb la DSP per tal de crear un control d'usuari o operari més adient, però els passos detallats per aconseguir-ho són:

Elegir un microcontrolador o la plataforma equivalent que ens permeti, mitjançant altres mòduls, comunicar i visualitzar de la forma més senzilla possible, estalviant feina en disseny de software i hardware, i si és possible costos de material.

Escollir els mòduls de comunicació i visualització, que realitzin la funció de forma correcte i que poden ésser connectats i programats de la forma més senzilla possible a la nostra plataforma escollida.

Una vegada tenim el hardware escollit a disposició, el següent pas és comprovar el funcionament de cada mòdul per separat, emprant petits programes que siguin semblants, a petita escala, del que necessitem a l'hora de connectar amb la DSP, per exemple amb el mòdul de comunicació realitzar petits programes que comuniquin amb el PC dades i tipus de dades que utilitzaríem de forma semblant amb la DSP.

Un cop tot llest, només manca anar dissenyant el software o programa principal de la plataforma realitzant ja petites proves amb la DSP, per tal de polir possibles errors o problemes.

Finalment, quan el programa principal ja funciona i per tant tenim una bona comunicació i visualització de dades amb la DSP de forma física, podríem esmentar millores e implementar-les si és possible.

1.3.Estructura del treball

L'estructura d'aquest projecte està pensada per conduir al lector o usuari, per les diferents etapes o apartats, d'integració i disseny de la plataforma en consonància amb els objectius a consolidar vistos anteriorment, per tal d'introduir la temàtica de la forma més coherent i simple possible.

El següent capítol, després d'aquesta introducció, es realitza un petit estat de l'art dels inversors que es troben actualment a la venda i les seves característiques. També s'intenta preveure les possibles tendències de futur en aquests dispositius.

Un cop vist l'estat actual dels inversors, passem a veure la descripció de la plataforma al complert que s'ha escollit per aquest projecte i la justificació de la mateixa.

En els següents quatre capítols, es detalla el gruix de la feina realitzada en aquest projecte, es descriuen els dos mòduls i com interactuen amb la plataforma escollida i la DSP. S'exposa el programa complert i els resultats obtinguts quan realitzem la connexió física amb la DSP.

Per acabar de consolidar el projecte, s'exposen les conclusions del que s'ha realitzat i els resultats obtinguts, així com els problemes i les solucions aportades.

2.Estat actual dels inversors

2.1.Control·ladors de motor actuals disponibles, inversors.

En aquest apartat es realitzarà un resum de les característiques del panell d'operari dels inversors, a la venda actualment i al final analitzarem les diferències amb aquest projecte.

En els inversors actuals, el panell d'usuari es troba dintre de la mateixa caixa on s'ubica la DSP i la electrònica de potència, mitjançant, en gairebé tots els casos, un LCD de dues línies de text i els botons necessaris per tal de realitzar les accions d'operari més simples. Aquests panells són completament tancats en el context del firmware i comunicació, e incorporen tecnologia de fa més de 20 anys, ja que es pot entendre que les marques no veuen necessari realitzar una millora en tecnologia en un producte industrial. En el capítol de la bibliografia, hi ha un link [COMDSP-1], a una pàgina web d'un proveïdor d'inversors de les marques més importants mundials i on es basa la nostra cerca en aquest apartat dels diferents models.

A continuació, analitzarem el panell d'usuari que incorporen els següents inversors a la venda actualment:

-Inversor MX2 de la marca OMRON:

Aquest inversor senzill, com podem veure a la figura 3, incorpora un

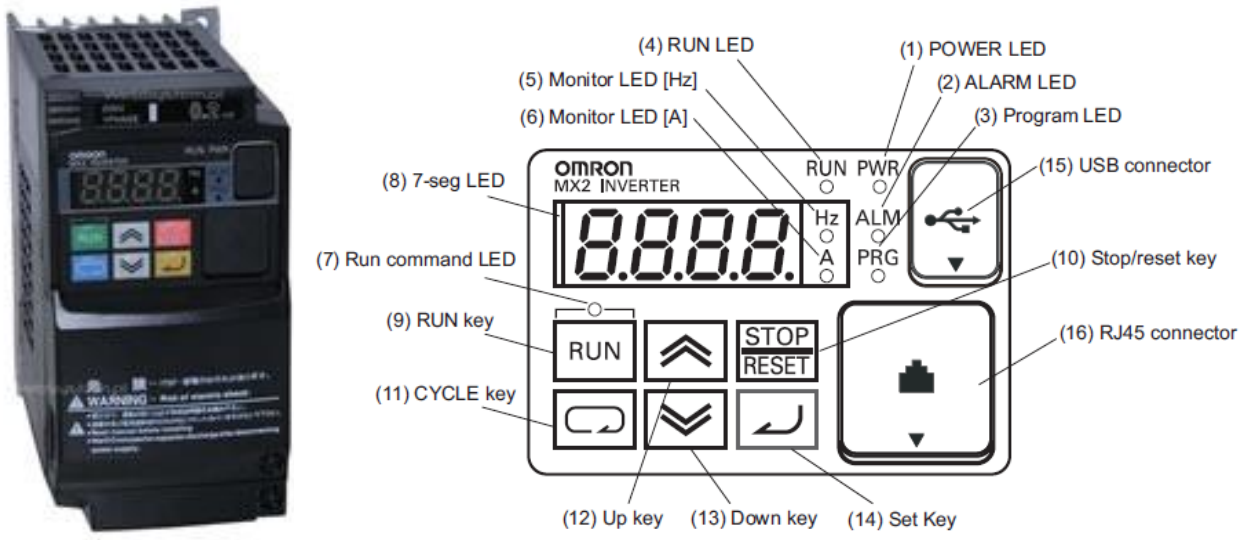


Figura 3: Inversor MX2 OMRON i descripció del panell d'usuari.

tipus de visualització formada per 4 displays de 7 segments amb els accionaments i botons necessaris. En la bibliografia s'ha inclòs el link al manual d'usuari [COMDSP-2], on s'expliquen els procediments per realitzar el control amb el panell d'usuari i la possibilitat de connectar un PC al mateix i modificar paràmetres dintre d'uns marges.

El panell d'usuari permet escollir entre diferents tipus de funcions i paràmetres a aplicar al motor, ja programades i només es poden canviar aquestes funcions connectant un PC al connector USB que es mostra a la figura 3. Podríem dir que el panell es força senzill i no hi ha la necessitat o el interès en millorar aquest dispositiu. Com a exemple de procediment d'interacció amb l'inversor, a la figura 4, es mostra com aquest panell configura el nombre de pols del motor a connectar, les accions a realitzar amb els botons i la visualització amb el display.

Action	Display	Func./Parameter
(Starting point)	bO 12	Level of electronic thermal setting
Press the [RUN] key	H001	"H" Group selected
Press the [▲] key three times	H004	Motor poles parameter
Press the [ENT] key	H004	2 = 2 poles 4 = 4 poles (default) 6 = 6 poles 8 = 8 poles 10 = 10 poles
Press the [▲] / [▼] key to select	H004	Set to your motor specs (your display may be different)
Press the [ENT] key	H004	Stores parameter, returns to "H004"

Figura 4: Configuració del nombre de pols del motor a connectar a l'inversor MX2.

-Inversor MicroMaster 440 de SIEMENS:

En aquest cas tenim un inversor molt més complex que l'anterior, ja que realitza moltes més funcions com a inversor i també millora el panell d'operari (dispositiu AOP), mitjançant un LCD de tecnologia TFT, on aquest permet el multi-llenguatge, connexió de més d'un inversor, visualitza molts més paràmetres a l'instant, etc., tal com es mostra a la figura 5.

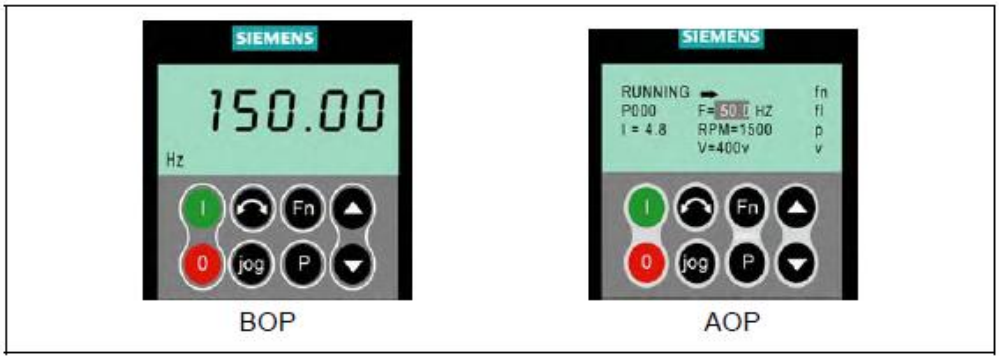


Figura 5: Inversor Micromaster 440 de SIEMENS + panell d'usuari opcional AOP.

Com es pot veure, aquest panell disposa d'una millor interacció amb l'operari, ja que disposa de més funcionalitat que el display de dígits, per exemple permet desar dades que amb l'altre és impossible de realitzar i disposa d'una touchscreen, així com programació d'alguns paràmetres que difícilment es poden realitzar en un display de dígits. Per acabar amb aquest inversor, també s'ha de comentar que incorpora una comunicació més oberta, al disposar de ports de comunicació RS232, CANbus i RS485, i per tant podem prescindir del seu panell d'operari e instal·lar-ne d'altres. L'accés al manual d'usuari, es troba a la bibliografia [COMDSP-3] amb un link de pàgina web.

Amb aquests dos inversors, ja tenim prou per arribar a la conclusió del que hi ha al mercat i quins tipus de panell d'usuari existeixen, ja que cercant altres fabricants i models, podem veure que tots ells són molt semblants.

2.2.Tendències de futur.

No és gens fàcil, per no dir impossible, trobar informació dels futurs dissenys i en que estan treballant les gran marques ara mateix respecte la innovació dels seus productes, segurament perquè serà un camp estratègic de cada companyia i no es vol difondre, però en aquest apartat veurem dues pistes que poden definir el futur dels panells d'operari, no només en l'àmbit dels inversors si no de tota la indústria de producció.

Com a primera pista, tenim un inversor de la marca parker que s'apropa una mica més a la plataforma d'aquest projecte i per tant és un primer pas o tendència. Estem parlant de l'inversor model AC30, on la única millora que podem destacar respecte el siemens micromaster 440, vist a l'anterior apartat, és que el panell de control d'operari es pot ubicar de forma remota a qualsevol lloc amb un kit (es desconeix quina tecnologia wireless utilitza, encara que el més segur és que sigui wifi). A la figura 6 es mostra l'inversor AC30 amb el panell d'operari i a la bibliografia hi ha un link amb la informació d'aquest de la companyia [COMDSP-4].

Figura 6: Inversor i panell d'operari de la marca Parker model AC30

Pot semblar no gaire important que un panell d'operari, mitjançant tecnologia wireless, es mogui per qualsevol lloc dins d'uns marges de distància però si ho serà quan analitzem la següent pista i cap a on sembla ser que van les tendències de futur.

La segona i última pista que sembla indicar les tendències de futur, és la creació d'empreses que es dediquen a realitzar un nou tipus d'automatització industrial, on tot dispositiu està connectat entre si amb una comunicació oberta i de diferents marques, inclús hi ha empreses que utilitzen hardware open source com Opiron [COMDSP-5]. Per tal de corroborar aquesta tendència, també tenim articles de publicacions especialitzades i orientades a la industria com "El futuro de la producción" de interempresas.net [COMDSP-6], on es parla clarament d'aquests nous processos industrials, i en resum es tracta que qualsevol màquina de caire industrial estigui connectada amb un sistema que automatitza la producció de forma conjunta amb altres màquines, i per tant que les industries del futur siguin el més modulars possibles i per tant més flexibles a l'hora de produir.

I com podem extrapolar l'anterior pista al cas dels inversors i els seus panells d'operari?, doncs com es pot veure en general, la màquina individual com la coneixem avui dia i el seu control individual, segons la tendència, ha de passar a un sistema més automatitzat i modular, on els panells d'operari seran únics per treballador i podran realitzar més feina de gestió i manteniment que no la tipologia de producció actual e individual i el dia a dia de la mateixa.

Per finalitzar el capítol, en el context d'aquest projecte final, amb unes poques modificacions hardware i software, es poden realitzar aquestes tendències de futur que s'han vist, ja que compleix amb un dels criteris que es veurà més endavant, l'únic inconvenient per obtindre un panell d'operari "universal" es resoldre amb les diferents companyies un llenguatge de comunicació i protocol que siguin oberts, per tal de realitzar aquesta interconnexió de màquines, aquest aspecte també obre la porta a crear noves empreses d'aquest tipus.

3.Descripció de la plataforma

3.1.Diagrama de blocs del sistema.

Con ja es van definir els objectius a consolidar, el diagrama de blocs del sistema, mostra de forma gràfica la connectivitat d'aquest projecte amb la DSP de l'inversor. A continuació es mostra a la figura 7, el diagrama de blocs d'aquest projecte.

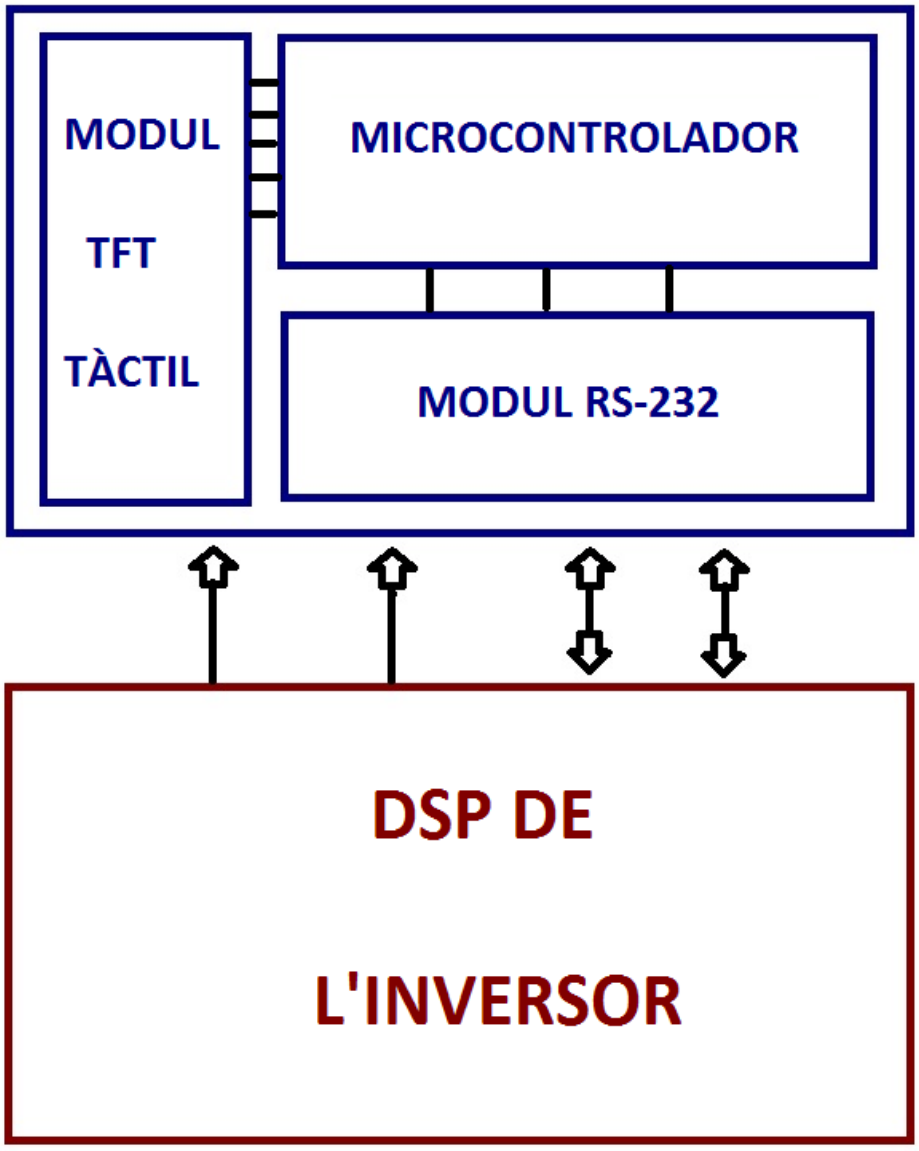


Figura 7: Diagrama de blocs del projecte, en blau fosc al plataforma d'aquest projecte, en vermell fosc la DSP de l'inversor i en negre les diferents connexions de forma simbòlica.

3.2.Introducció als elements

En aquest apartat, dos del objectius quedaran consolidats: l'elecció de la plataforma del microcontrolador i els dos mòduls connectats a la plataforma, el mòdul de visualització que està format per un display LCD de tecnologia TFT tàctil i el mòdul de comunicació RS232 que intercanviarà informació amb la DSP i la plataforma.

L'elecció de la plataforma amb microcontrolador, es realitzarà amb Arduino model UNO. La raó principal d'aquesta elecció ha estat el no haver d'escollir una plataforma tancada com per exemple els microcontroladors PIC on s'ha d'adquirir el programador per a la família de xip escollida, dissenyar e implementar el microcontrolador en una placa de circuit imprès adient i després buscar els diferents mòduls que tenen un cost més elevat i n'hi ha menys per escollir que amb l'Arduino, ja que els últims anys s'ha posat molt de moda; a més incorpora un entorn de programació anomenat IDE Arduino especialment dissenyat per aquest. Per tant, d'aquesta manera s'estalvia feina i diners en Hardware. A la figura 8, es mostra la placa Arduino Uno que s'utilitza en aquest projecte.

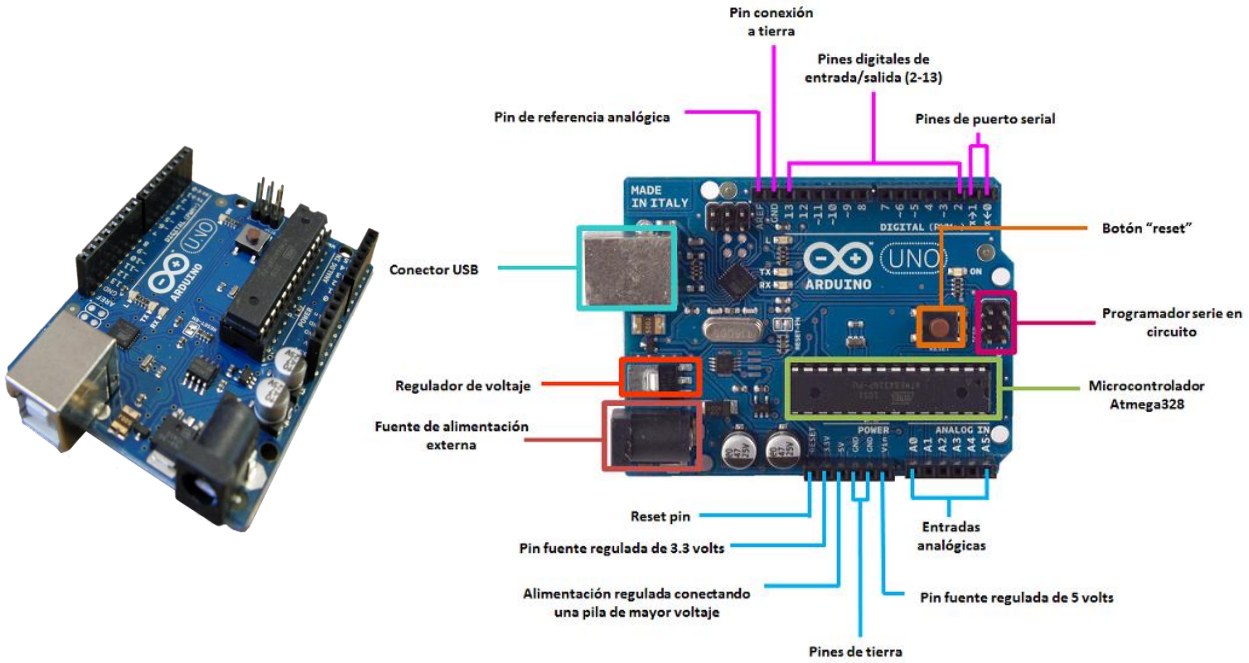


Figura 8: Placa del microcontrolador del projecte Arduino model UNO.

A continuació, es mostra una enumeració de les característiques més destacables d'aquest Arduino model UNO, a més a la bibliografia hi ha l'esquema de la placa [COMDSP-7] i el datasheet del microcontrolador de 8 bits que incorpora, que es tracta d'un Atmel model ATmega328P [COMDSP-8]:

- 14 terminals digitals d'entrada/sortida accessibles a la placa.
- 6 dels 14 terminals digitals anteriors es poden configurar com a sortides per PWM.
- 6 entrades analògiques amb un DAC de 10 bits cadascuna.
- Microcontrolador Atmel ATmega328P: 8 bits, 16 Mhz de rellotge, 32 KBytes de memòria flash, 1 KByte de ROM, 2 KBytes de RAM i arquitectura ARM.
- Programació mitjançant una connexió USB de tipus B accessible a placa.
- Alimentació directa per USB o mitjançant un connector d'alimentació accessible a la placa de 5 (v).
- Botó de reset accessible a la placa.

Com a exemple del software específic IDE de l'Arduino que es pot descarregar de la pàgina oficial [COMDSP-9], a la figura 9 es mostra el funcionament d'aquest software, executant un programa d'exemple "blink" que porta integrat als menús:

Figura 9: Programa d'exemple integrat a l'IDE de l'Arduino.

Com es pot observar a la figura anterior, el llenguatge de programació de l'Arduino, consta de dues funcions predeterminades, setup() i loop(), a setup() s'inicialitzen les variables i configuracions globals inicials del nostra programa, en canvi loop(), és una funció de bucle infinit que executa sempre el que hi ha dins les seves claus. El software permet crear altres funcions, però aquestes sempre han d'existir en qualsevol programa.

El programa de la figura, realitza la funció d'encendre i apagar un led que incorpora la mateixa placa d'Arduino, a la funció setup(), inicialitzem el pin digital número 13 de la placa com a sortida (pinMode (13, OUTPUT)) i a loop(), aquest pin número 13 es posa a nivell alt (digitalWrite (13, HIGH)) durant un segon (delay (1000)), es torna a posar a nivell baix (digitalWrite (13, LOW)) i es torna a esperar altre segon, i així s'anirà executant fins que retirem l'alimentació. Com veiem, realitzar un programa en la plataforma Arduino es força senzill, facilitant la feina en programes molt més grans.

L'elecció del mòdul de comunicació RS-232 es senzill d'escollir perquè en realitat només ha de tindre un circuit integrat MAX3232 [COMDSP-10] per convertir els nivells TTL de la USART disponible a la placa Arduino model UNO i els pins de connexió adients, tal com es pot veure a la figura 10, s'escull el mòdul RS-232 de l'empresa DF ROBOT [COMDSP-11], només utilitza quatre pins de la placa Arduino, el pin 0 de recepció RX, el pin 1 de transmissió TX, pin de massa o GND i el pin d'alimentació de 5 (v).

El pin número 0 i el 1, són els mateixos pins que utilitza el connector USB que desa el programa a la EEPROM de l'Arduino, per aquest motiu el mòdul incorpora un interruptor per tal de seleccionar el mode de programació USB o la comunicació RS232. A més incorpora un connector DB9 femella per connectar un cable sèrie, quatre terminals accessibles de tipus tira per accedir als quatre terminals que s'utilitza de la placa Arduino, un botó de reset que replica al de la placa Arduino, espai a la placa a mode de breadboard si volem implementar algun petit circuit i ens replica tots els ports de l'Arduino per si volem connectar una altra placa o mòdul a sobre per realitzar una altra aplicació (un aspecte molt important perquè a sobre es connecta el LCD-TFT).

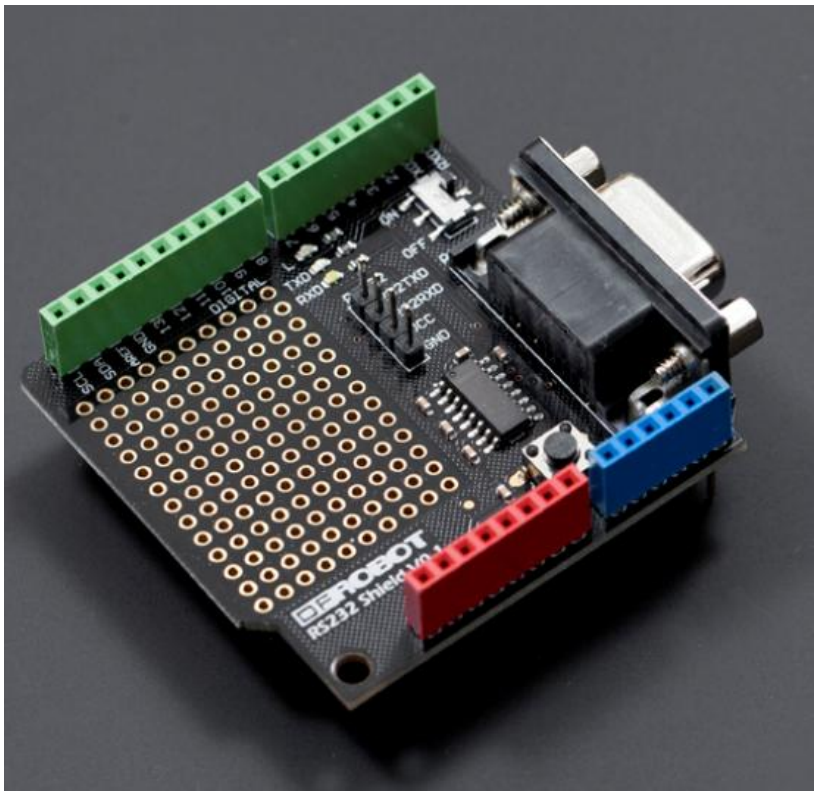


Figura 10: mòdul de comunicacions RS-232 per l'Arduino UNO de DF ROBOT.

L'elecció del mòdul de visualització LCD de tipus TFT tàctil, ha estat l'aspecte més complicat, finalment es va decidir utilitzar una placa de l'empresa Xinesa mcufriend [COMDSP-12], que es mostra a la figura 11.

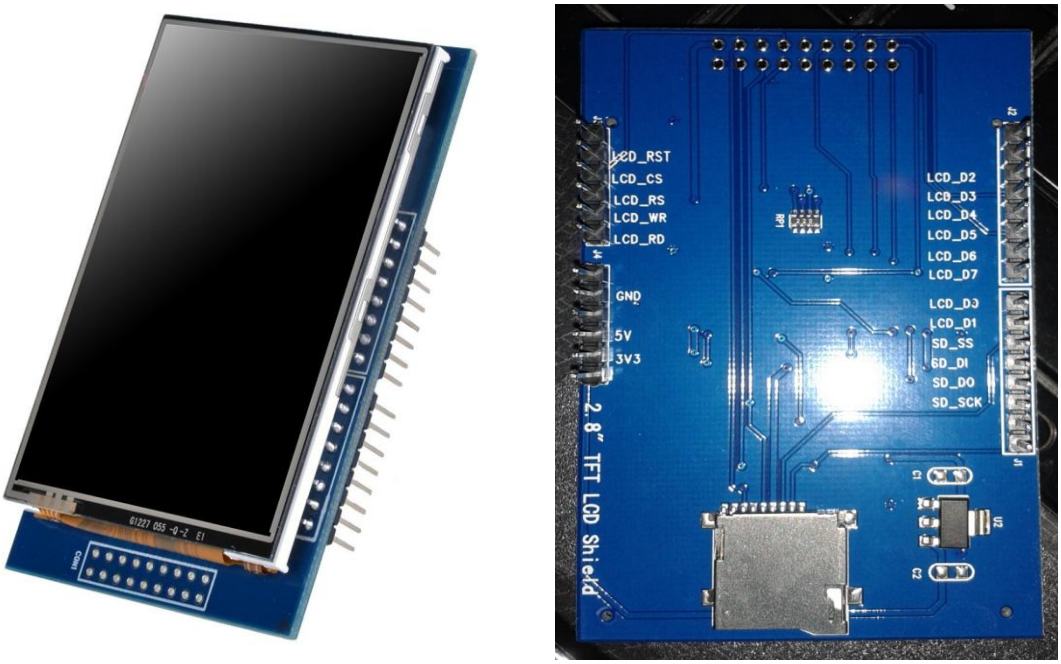


Figura 11: Mòdul LCD TFT tàctil de l'empresa xinesa mcufriend.

L'elecció d'aquesta pantalla es va realitzar buscant la millor eficiència possible, pel cost (2 o 3 cops inferior a proveïdors europeus) i per portar un driver de LCD conegut compatible amb Arduino, a més d' incorporar un panell tàctil en si mateix. Finalment la decisió no ha estat tant bona i s'explica al capítol 9 en l'annex de problemàtiques i solucions aplicades.

A continuació fem un resum de les característiques tècniques més importants d'aquest mòdul:

- Pantalla LCD gràfica TFT de 2,8" de 262000 colors.
- Xip controlador ILI9341 (segons fabricant) realment incorpora un controlador Sitronix ST7781 [COMDSP-13].
- Resolució de 320x240 píxels.
- Panell tàctil resistiu de 4 fils.
- Slot per a targetes de tipus micrSD amb un regulador de 3,3(v) a 2,7 (v). (No l'utilitzarem en aquest projecte).
- Connexió i comunicació en paral·lel compatible pin a pin amb Arduino UNO.

Com es pot apreciar a la figura 11, aquest mòdul utilitza els següents pins de l'Arduino:

- Pins d'alimentació: 5 (v), 3,3 (v) i GND.
- Entrades analògiques: A0, A1, A2, A3, A4 i A5 de la placa d'Arduino amb LCD_RD, LCD_WR, LCD_RS, LCD_CS i LCD-RST respectivament, a més la pantalla i la touchscreen utilitzen simultàniament dos pins analògics (A1 i A2 con YP (resistència vertical mínima) i XM (resistència horitzontal màxima) de la touccrenn).
- Terminals digitals: 2, 3, 4 i 5 de la placa d'Arduino amb SD_SCK, SD_D0, SD_DI, SD_SS, del slot microSD respectivament; 6, 7, 8, 9, 10, 11, 12 i 13 de la placa d'Arduino amb LCD_D1, LCD_D0, LCD_D7, LCD_D6, LCD_D5, LCD_D4, LCD_D3, LCD_D2; també la touchscreen utilitza a la vegada els pins digitals 6 i 7 (XP i YM, resistència horitzontal mínima i resistència vertical màxima, respectivament).

Per tant, gairebé tenim tots els terminals disponibles de la placa Arduino UNO ocupats; i entre el mòdul RS-232 i la pantalla no hi ha cap pin digital ni analògic compartit, així són compatibles respecte la connexió amb l'Arduino. Així ja tenim els dos primers objectius complerts, l'elecció de la plataforma amb microcontrolador i els mòduls de comunicació RS-232 i la visualització amb LCD de tipus TFT tàctil.

4.Descripció del funcionament i programació del mòdul RS232

En aquest capítol i els següents apartats, es detallarà el funcionament d'aquest mòdul, tant hardware com software i es comprovarà físicament mitjançant MatLab, abans de realitzar les proves amb la DSP i l'inversor.

4.1.Comunicació RS232 amb la DSP.

Abans de realitzar les proves, necessitem conèixer quin tipus de senyal necessita la DSP exactament, que surt per els pins RX i TX de la placa d'Arduino i com el mòdul RS-232 ens converteix el nivell del senyal.

Comencem per la DSP: exactament es tracta d'una DSP de la marca Texas Instruments model F28335, en la figura 12, es mostra l'esquema de la placa on va muntada i on es pot veure que la nostra connexió mitjançant RS-232 es realitzarà al integrat U5 que és un MAX3221, la mateixa família que tenim al mòdul DF ROBOT. Però necessitem un altre document per entendre quins valors surten de la DSP cap aquest xip i per tant que tindrem a la sortida, aquest document també és oficial de la Texas Instruments anomenat "TMS2833X Serial Comunications Interface SCI Reference Guide" [COMDSP-14].

A la pàgina 11 d'aquest document, s'explica que la DSP permet connectar altres dispositius mitjançant una comunicació asíncrona de tipus Non Return to Zero o NRZ, això significa que els nivells lògics tipus TTL es transformen a nivells de tensió positius i negatius sense passar pel valor de 0 (v), amb aquests valors de tensió transformats, el senyal pot arribar molt lluny i és el format que veurem més endavant que compleix amb el RS-232. També la DSP te la opció de configurar aquesta comunicació asíncrona amb processos d'integritat de la informació com per exemple: detecció de talls de comunicació, d'errors i paritat (mitjançant el registre SCICCR). A la figura 1-3 d'aquest document, apareix una taula que ens indica la trama d'enviament i recepció de dades amb aquest format i que és el que utilitzarem, aquesta taula es mostra a la figura 13.

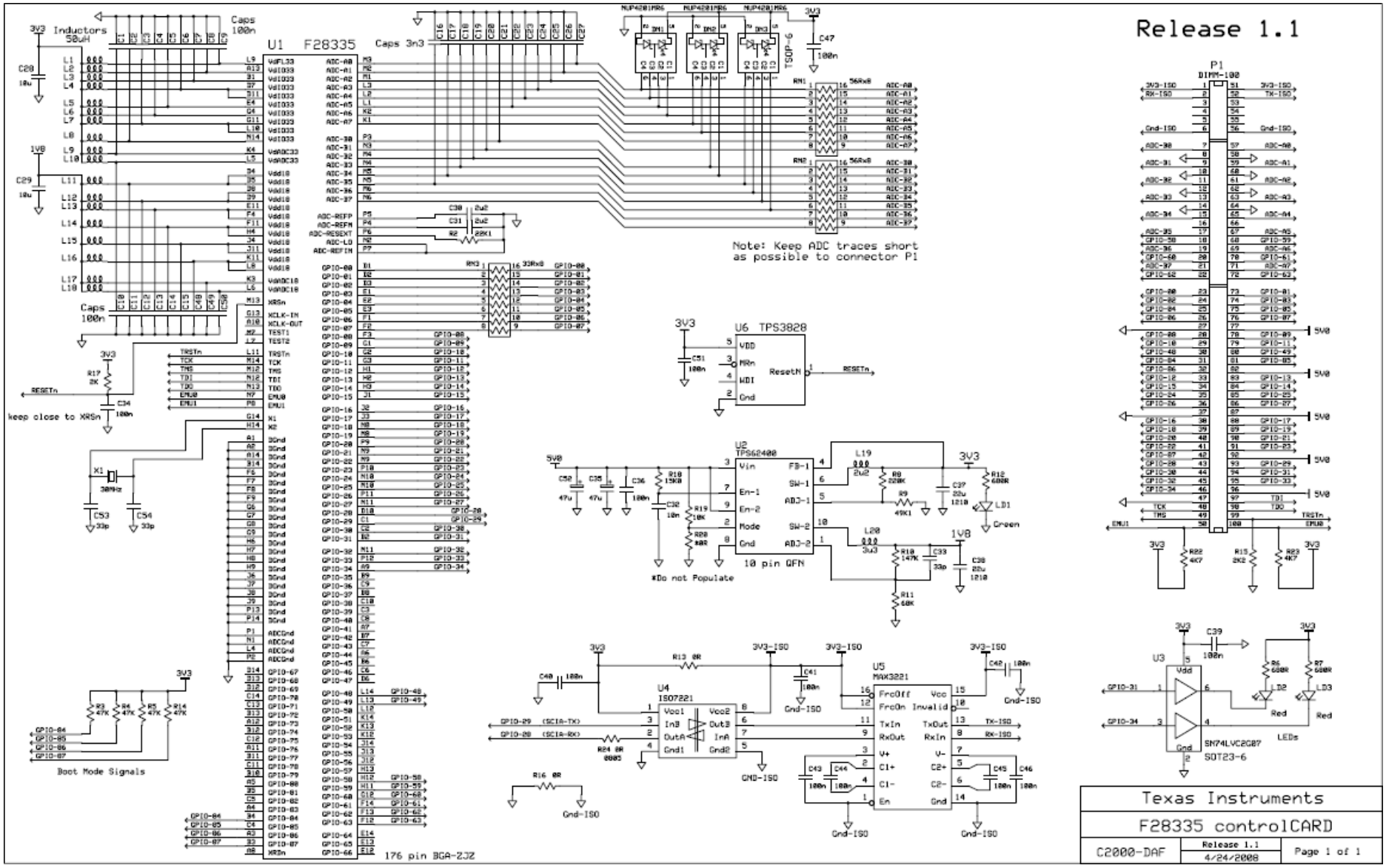


Figura 12: Placa de la DSP de Texas Instruments F28335

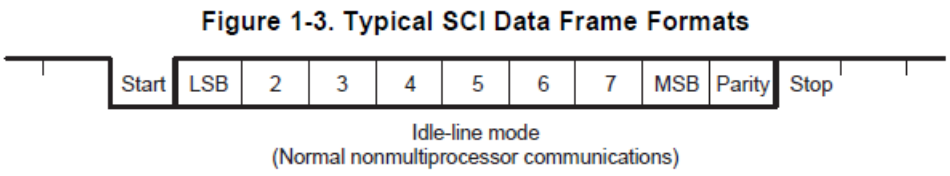


Figura 13: trama de dades de comunicació asíncrona de la DSP.

Per tant, ara coneixem quin tipus de senyal física demana la DSP per tal de comunicar-se correctament de forma asíncrona, en resum la DSP treu per els ports GPIO28 i GPIO29 (SCIA RX i TX respectivament, es veu a la figura 12), senyals digitals de tipus TTL (0 lògic i 1 lògic) amb una tensió d'operativitat de 3,3 (v). Per tal d'ésser compatibles en nivell de tensió amb els senyals NRZ, aquests dos ports es connecten al integrat U5 MAX2321, que transforma els nivells de tensió augmentant-los i transformant aquests a NRZ, on el valor lògic 0 tindrà tensió positiva i el 1 lògic tensió negativa. Representem aquest concepte de forma gràfica a la figura 14.

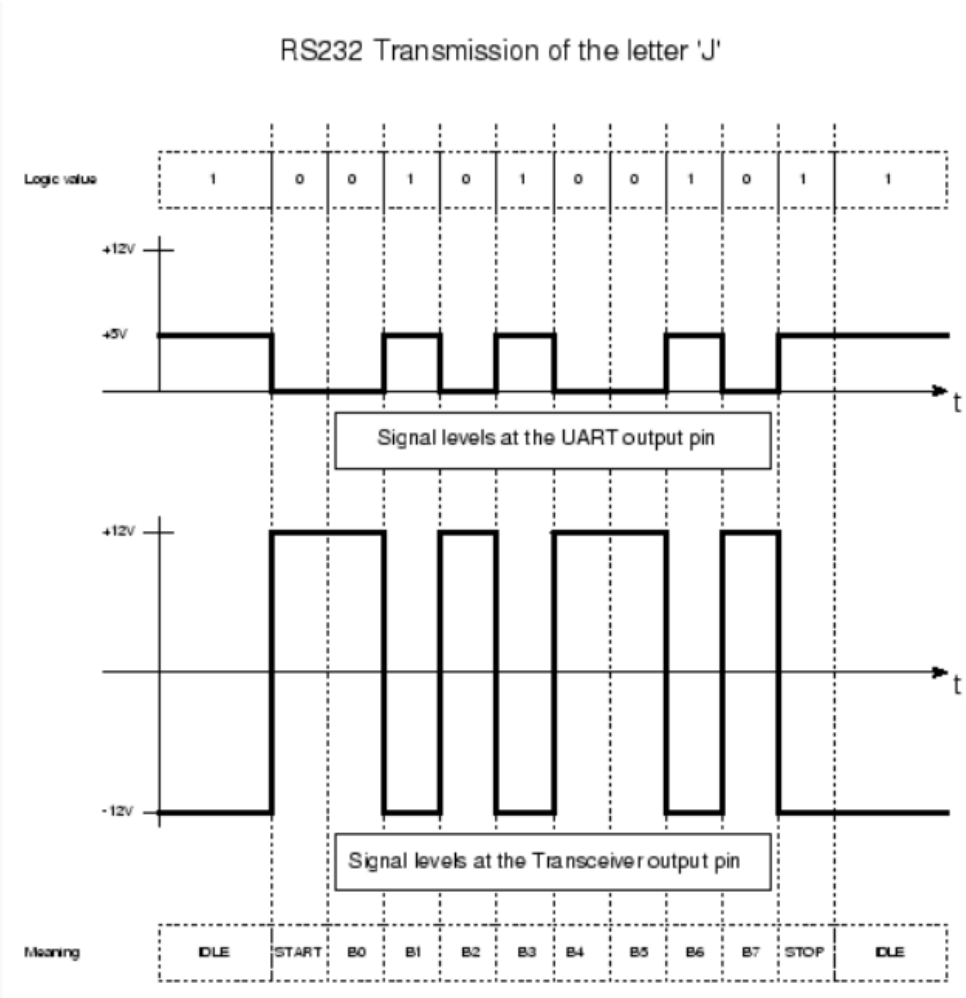


Figura 14: Sortida sèrie del caràcter "J" en format ASCII, a dalt nivells TTL/CMOS i a sota nivells NRZ RS-232.

La tensió que necessitem és exactament la que es mostra a la figura 14, per tant ara analitzarem quin tipus de senyal tenim amb la placa Arduino i el mòdul Rs-232 de comunicacions escollit.

A continuació, a la figura 15, es mostra un programa senzill d'Arduino que ens enviarà la lletra "A" en majúscula pel port TX de la placa (pin número 1):

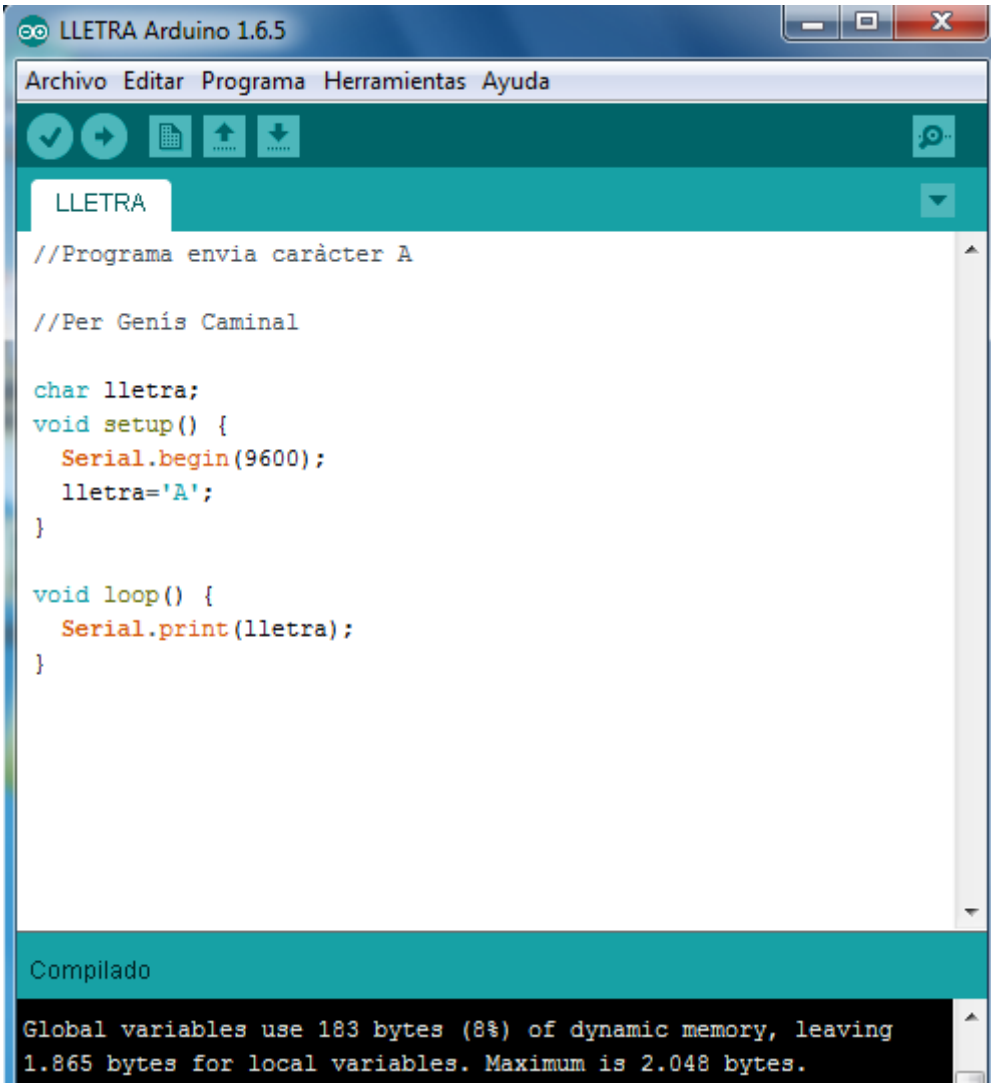


Figura 15: Programa senzill per enviar el caràcter A pel port TX o pin 1 de la placa d'Arduino.

Un cop tenim el programa llest sense cap error i connectat el mòdul RS-232, es recorda que anteriorment s'ha comentat que el mòdul RS-232 i la placa d'Arduino comparteixen el pin 0 i 1, ja que al pujar un programa a la memòria EEPROM del microcontrolador es realitza mitjançant aquests pins i el connector USB de la placa; per aquest motiu el mòdul RS-232 incorpora un interruptor, que abans de pujar el programa ha d'estar en OFF. Una vegada està pujat el programa, no cal que posem l'interruptor del mòdul a ON, perquè encara no realitzarem una connexió mitjançant

el RS-232, si no que mirarem amb l'oscil·loscopi que tenim al port TX o pin 1 de la placa d'Arduino i la sortida TXD del mòdul, així comprovarem que realment surt el caràcter enviat i les senyals coincideixen amb les del port de la DSP. A la figura 16, es mostra la captura d'aquests dos senyals amb un oscil·loscopi.

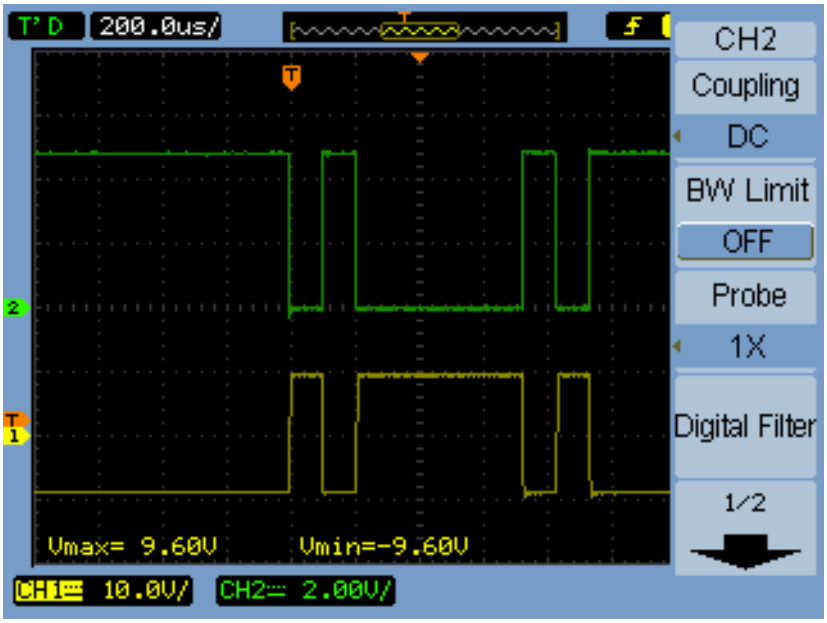


Figura 16: Captura pantalla d'oscil·loscopi, senyal del canal 2 de color verd, sortida TTL placa Arduino i l'altre senyal de color groc o canal 1, és la sortida del mòdul RS-232.

Com es pot apreciar en la captura, el senyal de color verd o canal 2 de la placa Arduino, al començament es troba per defecte en nivell alt i després ens envia el bit de Start que és un zero, després ens envia sencer el byte, i per últim torna a nivell alt que indica el bit de Stop, la informació enviada te els següents valors: 10000010 que es el mateix que: 1000 0010, com representa el codi ASCII, els primers 4 bits són els més petits (el primer zero equival a 1 en decimal) i els altres quatre, els més grans, com és un byte, el codi va des de 0 fins 255 i si passem a decimal cada grup de 4 bits tenim que 1000 = 1 i 0010 = 64, sumats dona 65 que en el codi ASCII és el caràcter A, per tant el programa és correcte. Ara només cal veure que en el canal 1 o senyal groc de la captura, tenim el mateix resultat, però invertit, a la sortida del mòdul RS-232. Aquesta inversió resulta dels nivells de tensió RS-232 i NRZ on el 0 lògic val 9,6 (v) i el 1 lògic val -9,6 (v) sense passar en cap moment per 0 (v), on es veu clarament la discontinuïtat; per tant el tipus de senyal que surt del mòdul és completament compatible en nivell de tensions i format al port de la DSP.

Per acabar de preparar la connexió del mòdul amb la DSP, cal veure la configuració de la trama que envia la placa d'Arduino per defecte, tal com mostra la figura 17.

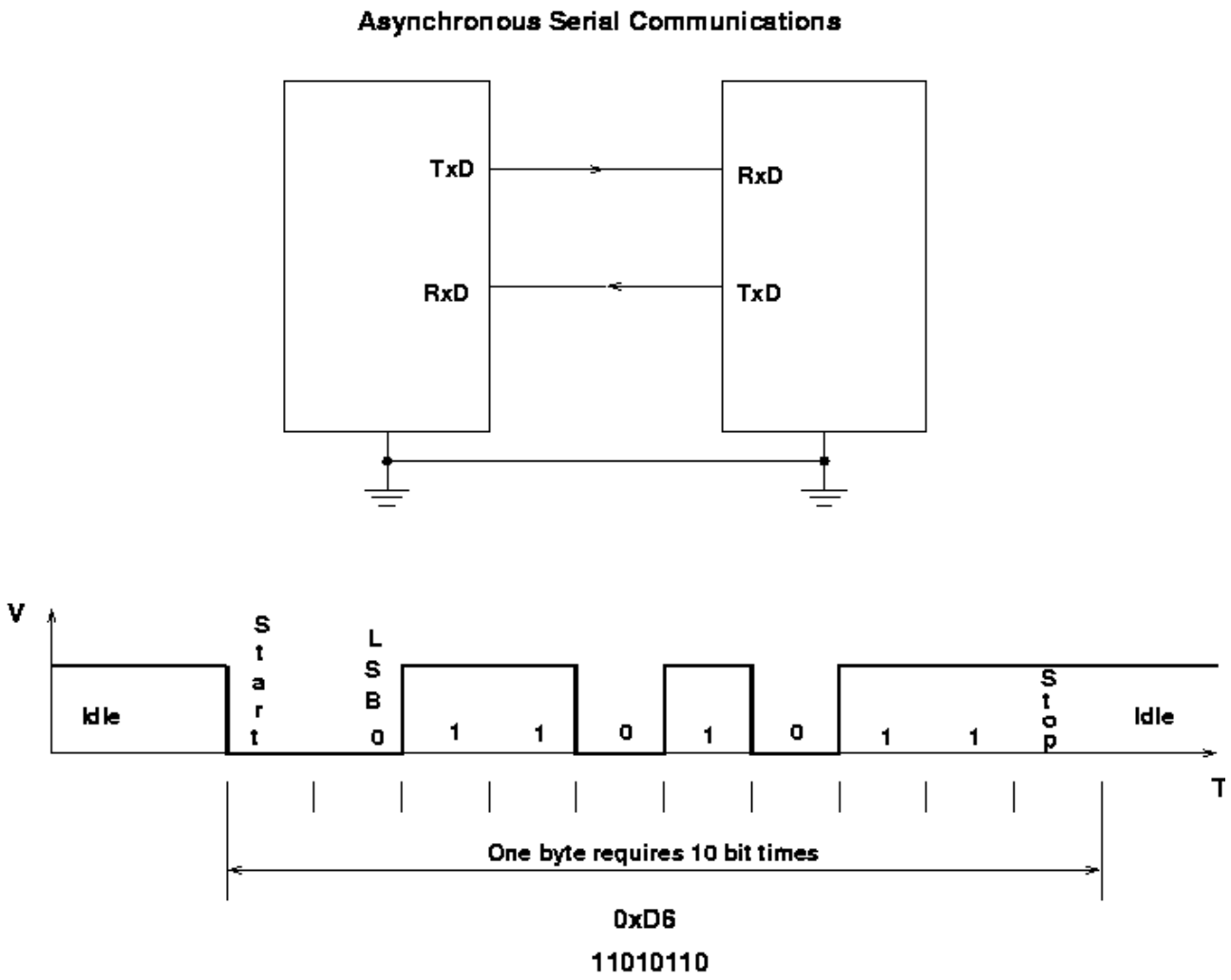


Figura 17: Enviament per defecte d'informació sèrie de la placa Arduino.

Es pot corroborar que la trama coincideix amb l'explicació anterior, però manca la paritat, això es degut a que per defecte s'envia sense paritat i en principi, en aquest projecte realitzarem les proves el més senzill possible i d'aquesta manera. També remarcar que en una connexió sèrie, els ports dels diferents dispositius a comunicar s'han de creuar i compartir la seva massa o GND, que és el que es pot observar a la part de dalt de la figura anterior.

4.2.Programació del mòdul i primeres proves mitjançant MatLab.

Per tal de comprovar a nivell de software l'enviament i recepció de les trames d'informació, utilitzarem el programari MatLab que ens ajudarà en el procés.

En aquest apartat, es comproven dos programes de dificultat diferent, per tal de poder desenvolupar el programa final del projecte.

Per tal de programar el mòdul, necessitem poques comandes, ja que a més de que l'Arduino utilitza el port USB per pujar programes, incorpora un monitor serial i algunes comandes pròpies, que seran les que utilitzem també amb el nostre mòdul. Per poder veure-ho millor, comencem amb el programa de prova, en aquest programa s'apagarà i encendrà un led que està incorporat a la placa d'Arduino quan des del port sèrie, l'Arduino rep el caràcter "a" o el caràcter "b" respectivament, a més es respondrà al MatLab amb els caràcters OFF i ON, també respectivament. EL programa complert, es mostra a la figura 18.

Figura 18: Programa que encén i apaga un LED i respon ON i OFF via port sèrie.

Les comandes que utilitzen el port sèrie es veuen fàcilment a la figura anterior, ja que tenen el prefix "Serial" que pertany a la funció del IDE de l'Arduino, a continuació l'explicació de cadascuna:

- Serial.begin (9600): aquesta funció inicialitza el port USB de l'Arduino amb una velocitat de 9600 Baudis.

- Serial.available (): aquesta funció detecta si hi ha informació al buffer d'entrada de l'Arduino, és important a l'hora de detecció de buffers en comunicació sèrie que en el cas de l'Arduino disposa de 64 Bytes de buffer d'entrada i sortida.

- Serial.read (): aquesta funció llegeix un sol caràcter d'entrada que correspon amb la trama de 8 bits d'informació que hem vist a l'apartat anterior.

- Serial.println (): aquesta funció envia el que hi ha entre els seus parèntesis, mitjançant caràcters ASCII, el text entre cometes o el valor de qualsevol variable. Després, finalitza amb un salt de línia ja que porta ln escrit al final, si no el volem podem escriure print i no hi haurà espai entre caràcters.

Per la resta del programa, podem veure que el LED que incorpora la placa està connectat al pin digital número 13, que l'inicialitzem com a sortida i l'anomenem led com a variable, s'inicialitza el port sèrie USB i llegim el caràcter d'entrada, que en els condicionals if, apaguem o encenem el led i responem OFF i ON depenent del caràcter rebut. A més aquestes accions s'executen indefinidament fins que deixem de rebre i enviar informació o deixem d'alimentar el microcontrolador donat que es troba dins la funció loop().

Un cop llest el programa, com el IDE i el llenguatge de l'Arduino incorpora un monitor sèrie, podem comprovar que funciona correctament com mostra la figura 19. En aquesta, es pot veure que al activar el monitor sèrie (es troba dins del IDE), ens hem de fixar que la velocitat que s'ha escrit dins el programa amb la funció Serial.begin (9600), sigui la mateixa que apareix en la part inferior del monitor (de 9600 baudis). En la part superior tenim un formulari on podem escriure text pla (codi ASCII) i fer clic a enviar per transmeteu al Arduino via sèrie, la resposta d'aquest es fa al formulari de sota on a la figura es pot veure que ja s'ha enviat una "b" i una "a" respectivament.

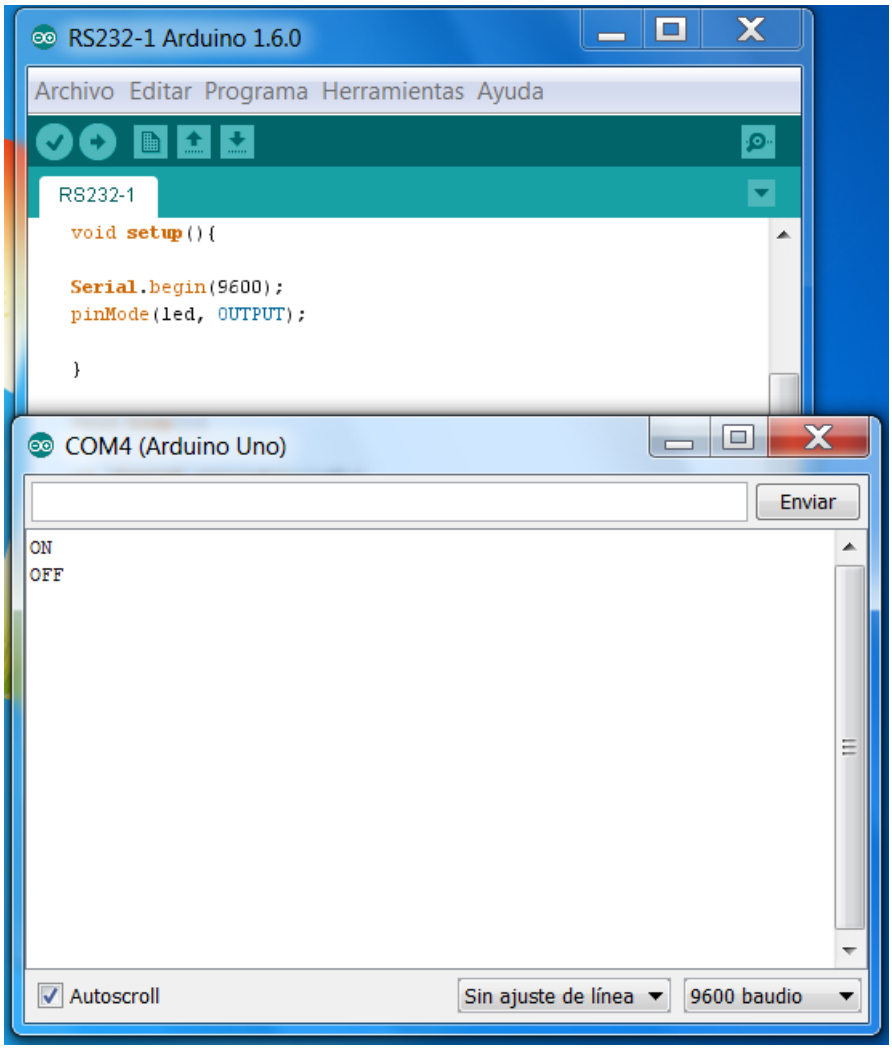


Figura 19: Monitor sèrie del IDE de l'Arduino executant el programa.

Ara que el programa funciona correctament, hem de dissenyar un programa a l'altre costat de la comunicació, que serà un PC amb el software MatLab com s'ha comentat anteriorment i col·locar el mòdul RS-232 damunt l'Arduino.

Un cop obert el MatLab, s'ha de configurar el port sèrie des del mateix software, cosa que és possible. Per fer-ho i no haver d'estar configurant el port en cada ocasió que l'obrim, es crea un fitxer de tipus*m que es quedarà desat a la zona del workspace del matlab. El codi de Matlab que ens configura el port és el següent:

```
clear all;
close all;
clc;
PS= serial ('COM4');
set (PS,'Baudrate',9600); %Velocitat de comunicació
set (PS,'StopBits',1); %Bit de parada a nivel lògic 1
set (PS,'Databits',8); %Trama d'informació de 8 bits
set (PS,'Parity','none'); %Sense paritat
set (PS,'Terminator','CR/LF'); %Caracter "CR o LF" finalitza la comunicació
set (PS,'OutputBufferSize',64); %Tamany del buffer de sortida
set (PS,'InputBufferSize',64); %Tamany del buffer d'entrada
set (PS,'Timeout',5); %Temps d'espera màxim per arribada d'informació
fopen(PS); % Habilita la comunicación del por COM4
```

Gràcies als comentaris (després del símbol %), es pot entendre tot el codi perfectament i es pot veure que te la mateixa configuració de trama que el port sèrie de l'Arduino. Aquest codi, el desarem amb el nom de PS.m i d'aquesta manera a cada programa que executem només caldrà posar PS al començament per tindre el port inicialitzat i disponible.

Un cop tenim el port, anem a veure el codi del programa dissenyat per a que realitzi les funcions adients:

```
PS
for i=1:10
fwrite (PS,'a','uchar');
k1=[];
for j=1:3
k1=[k1 fread(PS,1,'uchar')];
end;
k1=char(k1);
pause(1);
fwrite (PS,'b','uchar');
c1=[];
for z=1:2
c1=[c1 fread(PS,1,'uchar')];
end;
c1=char(c1);
pause(1);
end;
```

Com es pot veure, primer cridem l'arxiu PS.m anterior, ara que tenim el port inicialitzat, podem enviar dades. Després tenim un bucle for que repeteix deu vegades tot el programa, primer s'envia el caràcter "a" amb la instrucció fwrite que escriu al port sèrie PS, després amb un altre bucle for llegim la resposta de l'Arduino que seran els caràcters OFF, per això es crea un array de tres posicions perquè l'Arduino ens enviarà un caràcter per trama o cada cop, un cop omplert l'array,

tenim la instrucció k1=char(k1) que ens tradueix l'array de codi ASCII al nostre alfabet. Acte seguit, es fa una pausa d'un segon (això és excessiu però estem realitzant proves encara i es pot eliminar perfectament), i es repeteixen les mateixes funcions però pel caràcter "b" i un altre array més petit de dos posicions per rebre els caràcters ON corresponents de l'Arduino.

Una vegada tenim el codi, ja podem connectar l'Arduino al PC mitjançant el mòdul RS-232 també connectat. La connexió s'ha realitzat amb un cable creuat (com es mostrava a la figura 17) soldat a un connector db9 i amb una tira connectada al mòdul de comunicació de l'Arduino, ja que no s'ha trobat aquest tipus de cable als comerços per ser un tipus de connexió antiga, a més el PC on s'han realitzat les proves, no disposa de port sèrie db9 i s'ha comprat un adaptador USB-RS232, a la figura 20 es mostra l'Arduino amb el mòdul RS-232 connectat amb el cable; i la fotografia de l'adaptador USB-RS232. Un cop tot connectat, l'Arduino i el mòdul RS-232 estan esperant rebre la comunicació i només ens manca executar el codi del MatLab per a que funcioni.

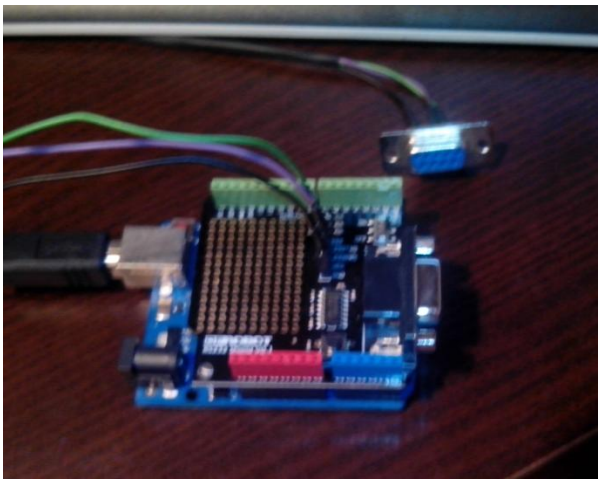


Figura 20: connexions mòdul Arduino RS-232 i adaptador USB-RS232 per al PC.

Ja executat el programa, a la figura 21 es mostra el resultat de l'array c1 ple amb els caràcters de resposta que ha enviat el mòdul RS-232 connectat a l'Arduino. A la figura 22, es mostra el mòdul RS-232 en funcionament.

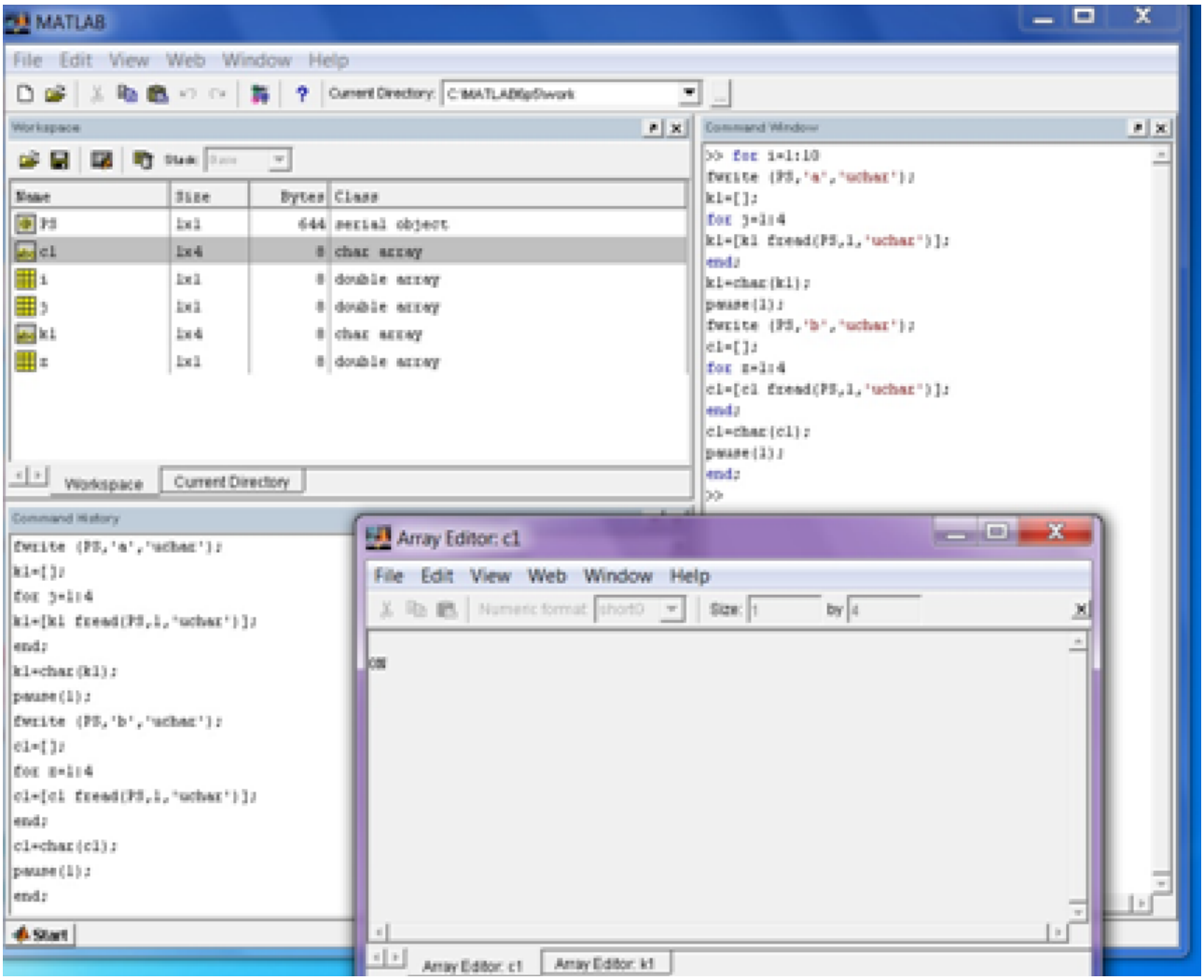


Figura 21: Array c1 del programa MatLab executat ple amb els caràcters rebuts de l'Arduino mitjançant RS-232

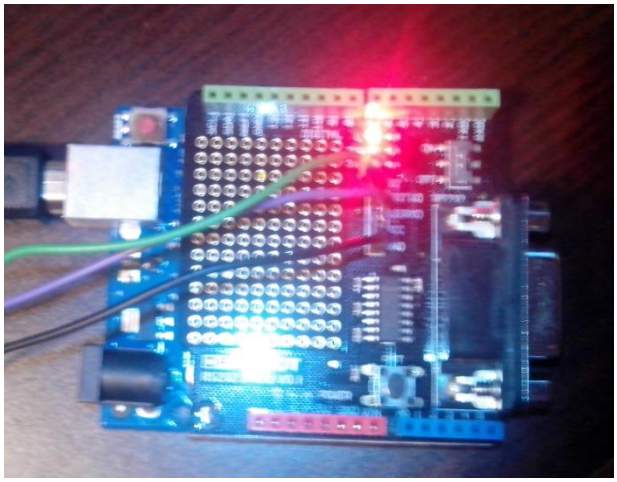


Figura 22: Mòdul RS-232 en funcionament connectat a la placa Arduino on també es pot apreciar la connexió del cable creuat.

Degut a que necessitem treballar amb números de tipus enter i la possibilitat també de que siguin de coma flotant, la comunicació anterior, no ens serveix per realitzar-ho, per tant la següent prova te aquest objectiu. A continuació, en la figura 23, es mostra el codi d'Arduino de la següent prova:

RS232-2

```
//Millora exemple RS232-2 enviant nombres enters i de coma flotant

//Per Genis Caminal

   int option;
   int led = 13;
   int velocidad=0;
   float velflo=0;

   void setup(){

   Serial.begin(9600);
   pinMode(led, OUTPUT);

   }

   void loop(){

   if (Serial.available()>0){
   option=Serial.read();
   if(option=='a') {
   digitalWrite(led, LOW);
   Serial.print("OFF");
   for (int i=1; i<10; i++) {
    Serial.print(velocidad);
    velocidad=velocidad+1;
   }
   }
   if(option=='b') {
   digitalWrite(led, HIGH);
   Serial.print("ON");
   for (int k=1; k<10; k++){
       Serial.print(velflo);
       velflo=velflo + 0.25;
       }
   }
 }
 velocidad=0;
 velflo=0;
}
```

Figura 23: Codi Arduino segon exemple mòdul RS-232.

Com es pot veure a la figura, el codi és una extensió semblant a l'exemple anterior, tenim un altre cop dos condicionals que apaguen el led de la placa quan es rep el caràcter "a" i encenen el led quan es rep el caràcter "b" de del port sèrie mitjançant el mòdul RS-232, respectivament, però dins d'aquests condicionals tenim: quan rebem el caràcter "a", dins d'un bucle for, s'incrementa un enter i s'envia pel port sèrie, i així fins 9 vegades, és a dir, s'envia una llista de 9 enters des de 0 fins a 8; i quan rebem el caràcter "b", dins del bucle for, s'incrementa un número de coma flotant o irracional i s'envia pel por sèrie, fins 9 vegades, és a dir s'envia una llista de

9 floats des de 0.0 fins a 2.0. I com l'exemple anterior, es va repetint mentre enviem informació o tallem l'alimentació.

A la figura 24, es mostra el resultat del monitor sèrie de l'IDE de l'Arduino.

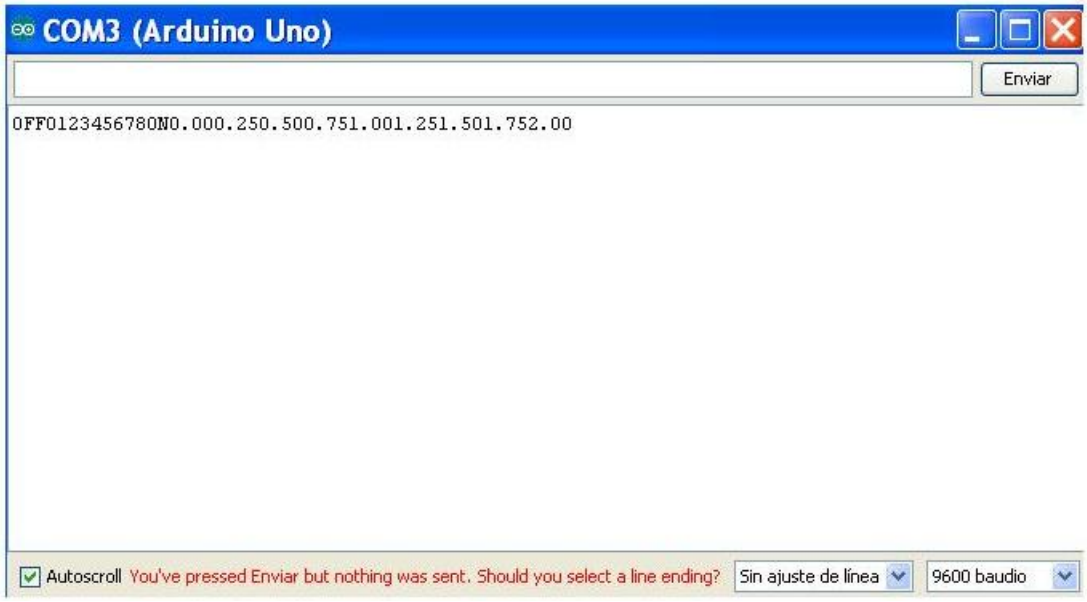


Figura 24: Resultat amb el monitor sèrie de l'Arduino.

Com es mostra a la figura 24, tenim el resultat d'enviar el caràcter "a" primer i el resultat d'enviar el caràcter "b" després, sense espais. El resultat es correcte, encara que la comunicació només es realitza entre el PC i l'Arduino.

Un cop vist que funciona el codi de l'Arduino, a conitnuació es mostra el codi del MatLab per tal de provar el funcionament del mòdul RS-232:

```
PS
PS.ReadAsyncMode='continuous';
pause(1);
fwrite (PS,'a','uchar');
k1=fscanf (PS,'%c',3);
for i=1:9
c(i)=fscanf(PS,'%d',1);
end
fwrite(PS,'b','uchar');
x=fscanf (PS,'%c',2);
for k=1:9
z(k)=fscanf (PS,'%f',4);
end;
```

Com es pot apreciar, aquest codi de MatLab és molt semblant a l'anterior exemple, l'únic canvi significatiu és que després d'enviar els caràcters corresponents "a" i "b", tenim dos bucles for que es repeteix 9 vegades, exactament de la mateixa

manera que teníem al codi d'Arduino. El primer bucle for, mitjançant la instrucció de MatLab `c(i)=fscanf(PS,'%d',1);` ens permet desar, en la posició i de l'array c, els valors tipus enter, amb aquest símbol %d, que troba al buffer d'entrada del port sèrie PS, i d'un en un, per aquest motiu hi ha un 1 al final del parèntesi. Per tant aquesta instrucció, gràcies al bucle for, ens desa 9 enters en cada posició de l'array c. El mateix passa amb l'altre bucle for `z(k)=fscanf (PS,'%f',4);`, on trobem l'array z, de 9 posicions k, marcat a les iteracions del bucle, que escaneja el port sèrie PS, en busca de números de coma flotant, gràcies al símbol %f i amb 4 digitis de tamany cadascun, incloent el punt de separació decimal.

El resultat final dels dos array omplerts, el tenim a la figura 25.

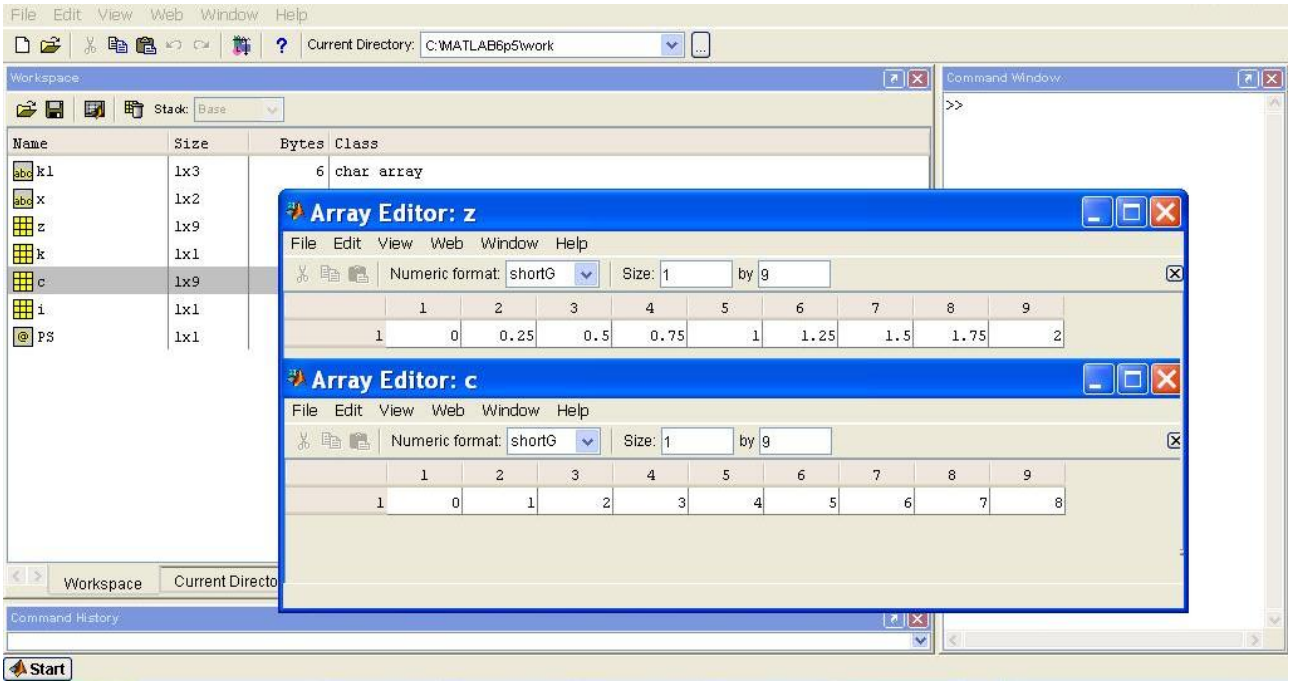


Figura 25: Arrays c i z omplerts amb els 9 enters i floats, mitjançant la comunicació PC-mòdul RS-232 de l'Arduino.

Amb aquests dos exemples donem per acabat el capítol sobre la comunicació amb el mòdul RS-232 de l'Arduino. Les següents proves de comunicació ja es podran realitzar sobre la DSP.

5.Descripció del funcionament i programació del mòdul LCD-TFT tàctil.

En aquest capítol veurem el funcionament del mòdul del LCD-TFT tàctil amb més detall que quan es va descriure al capítol 3. Primer començarem realitzant un resum dels detalls més importants del driver que incorpora aquesta pantalla i la seva touchscreen resistiva, després es veurà mitjançant dos apartats, les funcions per detectar pulsacions i dibuixar a la pantalla des del IDE de l'Arduino respectivament. A la figura 26, es mostra una taula amb el detall de totes les connexions del mòdul amb l'Arduino i la descripció de cadascuna [COMDSP-15].

Arduino Pin	LCD Shield Pin	Use
3.3V	3.3V	Power
5V	5V	Power
GND	GND	Power
A0	LCD_RD	LCD Control
A1	LCD_WR TOUCH_YP	LCD Control / Touch Data
A2	LCD_RS TOUCH_XM	LCD Control / Touch Data
A3	LCD_CS	LCD Control
A4	LCD_RST	LCD Reset
D2	LCD_D2	LCD Data
D3	LCD_D3	LCD Data
D4	LCD_D4	LCD Data
D5	LCD_D5	LCD Data
D6	LCD_D6 / TOUCH XP	LCD Data/ Touch Data
D7	LCD_D7 / TOUCH YM	LCD Data / Touch Data
D8	LCD_D0	LCD Data
D9	LCD_D1	LCD Data
D10	SD_CS	SD Select
D11	SD_DI	SD Data
D12	SD_DO	SD Data
D13	SD_SCK	SD Clock

Figura 26: Taula connexions mòdul de la pantalla amb l'Arduino i descripció de cadascun dels pins.

5.1.Descripció driver TFT i de la touchscreen.

Com ja es va comentar anteriorment, el driver que mou els gràfics a dibuixar a la pantalla, és el Sitronix ST7781 [COMDSP-14] a nivell hardware, la connexió de la configuració amb la pantalla es realitza mitjançant 4 pins del xip del driver, que es connecten a l'alimentació o a massa per tal de configurar la comunicació del driver amb la pantalla, aquesta informació es troba a la pàgina 22 taula 6.2 del datasheet del driver, això es mostra a la figura 27.

6.2 Interface Logic Pin

<table>
<tr><th>Name</th><th>I/O</th><th>Description</th><th>Connect Pin</th></tr>
<tr><td>IM0~IM3</td><td>I</td><td>-Select the MCU system interface mode
<table>
<tr><th>IM3</th><th>IM2</th><th>IM1</th><th>IM0</th><th>MCU-Interface Mode</th><th>DB Pin Use</th></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>I80-system 16-bit interface</td><td>DB[17:10]
DB[8:1]</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>I80-system 8-bit interface</td><td>DB[17:10]</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>I80-system 18-bit interface</td><td>DB[17:0]</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>I80-system 9-bit interface</td><td>DB[17:9]</td></tr>
</table></td><td>DGND/VDDI</td></tr>
<tr><td>RESET</td><td>I</td><td>-This signal will reset the driver and it must be applied to properly initialize the chip.</td><td>MCU</td></tr>
<tr><td>/CS</td><td>I</td><td>-Chip select input pin and signal is active low.
-This pin can be permanently fixed "Low" in MCU interface mode only.</td><td>MCU</td></tr>
<tr><td>RS</td><td>I</td><td>-Display data or command selection pin in MCU interface.
RS ='1': display data or parameter.
RS ='0': command.
-If not used, please connect this pin to VDDI or DGND.</td><td>MCU</td></tr>
<tr><td>/RD</td><td>I</td><td>-Read enable in 8080 MCU parallel interface.</td><td>MCU</td></tr>
<tr><td>/WR</td><td>I</td><td>-Write operation enable pin in 8080 MCU parallel interface.</td><td>MCU</td></tr>
<tr><td>DB[17:0]</td><td>I/O</td><td>-An 18-bit paralled bi-directional data bus for MCU system interface mode
8-bit Interface: DB[17:10] is used
9-bit Interface: DB[17:9] is used
16-bit Interface: DB[17:10] and DB[8:1] is used
18-bit Interface: DB[17:0] is used
-If not used, please connect this pin to VDDI or DGND.</td><td>MCU</td></tr>
<tr><td>SW_EE</td><td>I</td><td>-To use extended command set, please connect this pin to VDDI.
-During normal operation, please let this pin open.</td><td>DGND/VDDI</td></tr>
<tr><td>FMARK</td><td>O</td><td>-Output a frame head pulse signal is used as synchronies MCU to frame rate
-If not used, Let this pin open</td><td>-</td></tr>
</table>

Note1. When /CS="1", there is no influence to the parallel interface.
Note2. "1" = VDDI level, "0" = DGND level.

Figura 27: Taula amb la informació de la configuració de les connexions per tal de comunicar la pantalla amb el driver.

La connexió del driver, es realitza amb els 4 pins anomenats IM0-IM3, i el nostre mòdul està configurat com la segona fila de la taula, és a dir una connexió en paral·lel de 8 bits (pins del 17 al 10 del driver) i per tal de visualitzar correctament els 262k, es realitza una conversió interna que es mostra a la pàgina 33 del datasheet del driver.

També en aquesta taula apareixen les connexions importants /RD, /WR, /RS, /CS i RESET o /RST, que serveixen per:

- /RD: Si està a zero la lectura de dades s'habilita. ("LCD READ")
- /WR: Si està a zero s'habilita l'escriptura. ("LCD WRITE")
- RS: Per defecte es connecta a nivell lògic 1 i habilita el dibuix a la pantalla. ("COMMAND DATA")
- /CS: Per defecte es connecta a nivell lògic 0 i habilita la selecció de xip, només te utilitat si hi ha més d'un xip que l'utilitzi. ("CHIP SELECT")

A la figura 28, es mostra de forma gràfica tota la connexió que s'ha d'utilitzar entre el driver del LCD-TFT i l'Arduino.

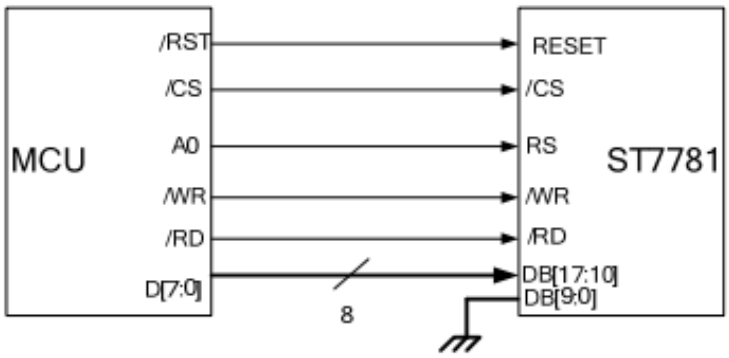


Figura 28: Connexió driver ST7781 i l'Arduino

Com es pot entendre, aquestes connexions ja les tenim realitzades al mòdul que hem escollit i només ens serà útil a l'hora de configurar el software.

La configuració de colors, tal com es comenta al datasheet pàgina 33, pot adreçar internament els 18 bits que són els 262K, encara que només ho fem amb 16 bits i tenim una connexió de 8 bits, aquest adreçament el realitza enviant dos bytes, el primer és la part alta i el segon és la baixa i la diferència de 2 bytes que existeix entre els 16 i els 18 bits el realitza el driver internament.

No ens esmentarem res més del driver, ja que tenim una idea de les connexions amb la pantalla i part del funcionament hardware; i la idea és realitzar les accions de dibuix el més senzill possible. Ara passarem a veure les connexions i el funcionament hardware de la touchscreen.

El driver ST7781, no te res a veure amb la touchscreen, de fet no deixa d'ésser una làmina resistiva superposada a la pantalla, que consisteix en dues capes separades de material plàstic conductor, amb una determinada resistència al corrent elèctric, que al pressionar un punt determinat de la capa exterior, fa contacte amb la inferior i mesurant la resistència calcula el punt exacte de la pressió . S'ha mesurat la resistència d'aquesta pantalla en repòs amb un multímetre entre els pins A2 i 6; i

s'ha mesurat una resistència de 365 ohms, sense pressionar cap punt de la mateixa. Com es pot veure a la figura 26, la touchscreen te una connexió amb la placa de 4 fils. Per entendre millor com funciona, a la figura 29, es mostra de forma gràfica el seu hardware.

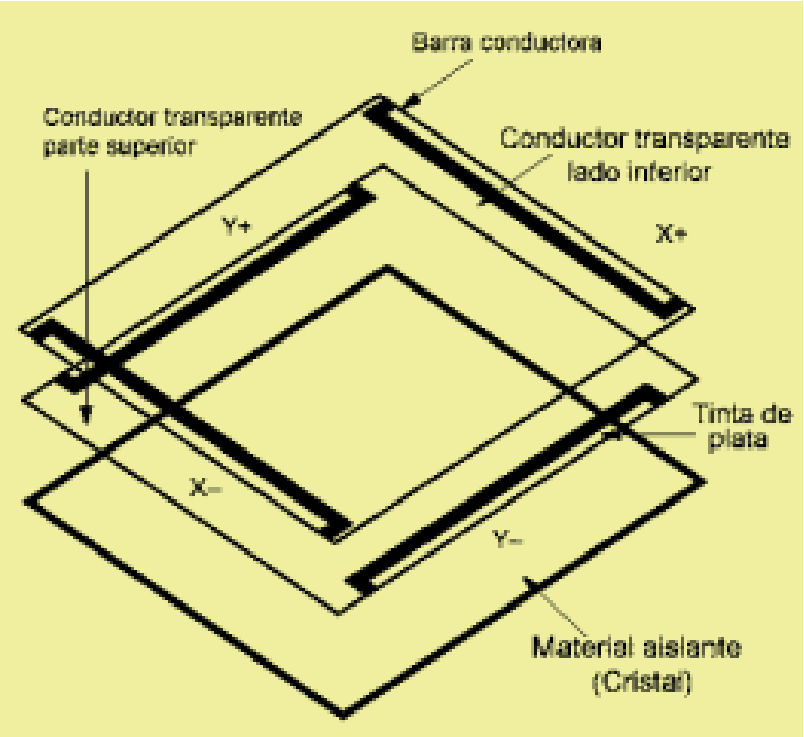


Figura 29: Hardware de la touchscreen.

Com podem veure a la figura, els quatre fils de connexió (XP o X-,YP o Y-,XM o X+ i YM o Y+), realitzen la funció de:

- XP o X-: Resistència mínima de l'eix horitzontal.
- YP o Y-: Resistència mínima de l'eix vertical.
- XM o X+: Resistència màxima de l'eix horitzontal.
- YM o Y+: Resistència màxima de l'eix vertical.
- Z (aquesta no apareix a la figura però la utilitzarem): és la pressió que s'extreu o calcula dels quatre paràmetres anteriors.

Per tant, es pot entreveure que tenim les coordenades X e Y i per tant al pressionar sobre la pantalla ens retornarà aquestes sobre el punt pressionat, i només detectarà un punt ja que aquesta tecnologia no permet dobles pulsacions.

Ara mateix ja tenim la informació de hardware suficient per tal de passar a cridar les funcions de software per programar el dispositiu, tal com veurem als següents apartats.

5.2.Funcions per detectar pulsacions a la touchscreen.

Per tal de detectar les pulsacions a la touchscreen, necessitem una llibreria que tradueix i calcula de nivell baix a alt, la pulsació realitzada. Aquesta llibreria s'anomena touchscreen del fabricant de material per Arduino Adafruit [COMDSP-16].

Un cop descarregada i descomprimida, per tal de poder cridar la llibreria en qualsevol programa, s'ha d'incloure a una carpeta determinada del IDE d'Arduino, la carpeta es troba dins el directori d'instal·lació, a la subcarpeta libraries, aquí enganxem la carpeta de la llibreria descarregada i descomprimida, tal com es mostra a la figura 30.

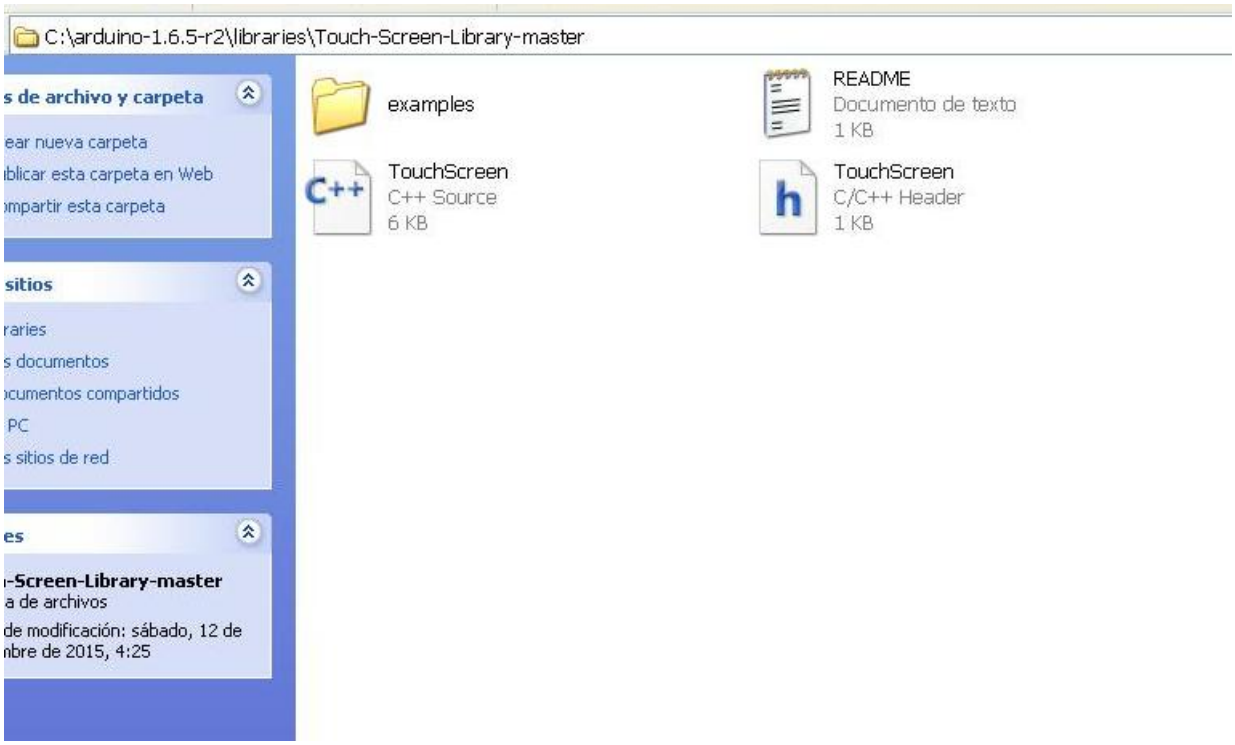


Figura 30: Adreçament i ubicació de la llibreria descarregada.

Amb aquest procés, ja podem cridar aquesta llibreria des de qualsevol programa del IDE i començar a utilitzar-la. Per tal de conèixer les funcions de software, a la carpeta d'aquesta llibreria, tenim uns exemples i executarem l'exemple touchscreendemoshield, que es pot veure des del IDE de l'Arduino, gràcies al ubicar la llibreria a la carpeta corresponent, tal com s'observa a la figura 31.

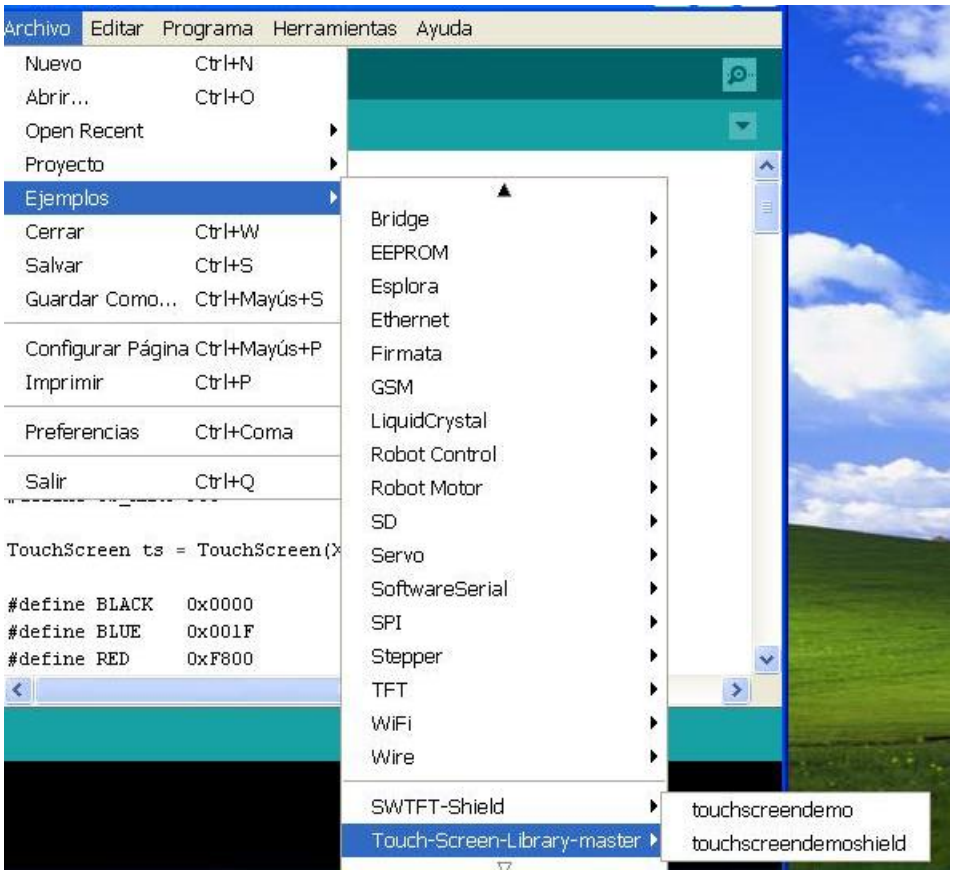


Figura 31: Exemples de la llibreria disponibles al IDE.

A continuació, es mostra el codi d'aquest exemple que llegeix contínuament la pulsació o pressió sobre la touchscreen de forma indefinida mentre es polsi:

```
#include "TouchScreen.h"

#define YP A1
#define XM A2
#define YM 7
#define XP 6

#define TS_MINX 120
#define TS_MINY 870
#define TS_MAXX 850
#define TS_MAXY 908

#define MINPRESSURE 200
#define MAXPRESSURE 1600

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 365);

void setup(void) {

Serial.begin(9600);
pinMode(13, OUTPUT);
```

```
}

void loop(void) {

digitalWrite(13, HIGH);
TSPoint p = ts.getPoint();
digitalWrite(13, LOW);
pinMode(XM, OUTPUT);
pinMode(YP, OUTPUT);


if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
   p.x = map(p.x, TS_MINX, TS_MAXX, 0, 239);
   p.y = map(p.y, TS_MINY, TS_MAXY, 0, 319);
   Serial.print("X = "); Serial.print(p.x);
   Serial.print("\tY = "); Serial.print(p.y);
   Serial.print("\tPressure = "); Serial.println(p.z);
}

}
```

A continuació es descriu pas a pas el funcionament del programa:

- **`#include "TouchScreen.h"`** : Aquesta instrucció ens inclou la llibreria touchscreen.h que ens permet utilitzar la touchscreen amb l'Arduino. Sempre és el primer pas, ja que si no s'inclou no funcionarà mai.
- **`#define YP A1; #define XM A2; #define YM 7; #define XP 6:`** Aquestes quatre instruccions, ens defineixen quins terminals de la placa d'Arduino, estan connectats els terminals de la touchscreen, tal com es va descriure a l'apartat anterior.
- **`#define TS_MINX 120:`** Aquesta instrucció defineix el punt mínim de resistència a la coordenada X, prèviament calibrada manualment.
- **`#define TS_MINY 870:`** Aquesta instrucció defineix el punt mínim de resistència a la coordenada Y, prèviament calibrada manualment.
- **`#define TS_MAXX 850:`** Aquesta instrucció defineix el punt màxim de resistència a la coordenada X, prèviament calibrada manualment.
- **`#define TS_MAXY 908:`** Aquesta instrucció defineix el punt màxim de resistència a la coordenada Y, prèviament calibrada manualment.
- **`#define MINPRESSURE 200:`** Aquesta instrucció defineix el marge mínim de pressió detectable per la touchscreenn, prèviament calibrada manualment.
- **`#define MAXPRESSURE 1600:`** Aquesta instrucció defineix el marge màxim de pressió detectable per la touchscreenn, prèviament calibrada manualment.
- **`TouchScreen ts = TouchScreen(XP, YP, XM, YM, 365):`** aquesta instrucció, insereix els terminals, definits anteriorment, i la resistència física sense pressió de la touchscreen a la funció ts, que ens permet detectar el

punt de pressió actual.

- **`Serial.begin(9600)`:** Aquesta instrucció habilita el port sèrie a 9600 Baudis de velocitat. En aquest programa el necessitem perquè veurem el resultat en el monitor sèrie del IDE.
- **`pinMode(13, OUTPUT)`:** Amb aquesta instrucció definim el terminal digital número 13 com a sortida i el necessitem per tal de detectar el punt més endavant.
- **`digitalWrite(13, HIGH)`:** Aquesta instrucció posa a nivell alt el terminal digital 13 de la placa Arduino, per habilitar la detecció del punt a la pròxima instrucció.
- **`TSPoint p = ts.getPoint()`:** s'anomena a p com la funció de tipus TSpoint (llibreria touchscreen), és a dir, al executar aquesta instrucció, detectem el punt pressionat en aquest moment, i es desa a p tres coordenades (x, y i z).
- **`digitalWrite(13, LOW)`:** Aquí es torna a deixar el terminal 13 a nivell baix, un cop llegit el punt pressionat.
- **`pinMode(XM, OUTPUT)`:** S'habilita el terminal X mínima com a sortida, per tal de llegir les dades.
- **`pinMode(YP, OUTPUT)`:** S'habilita el terminal Y màxima com a sortida, per tal de llegir les dades
- **`if (p.z > MINPRESSURE && p.z < MAXPRESSURE)`:** Condicional que ens limita la detecció del punt pressionat, és a dir, es realitzaran les instruccions dins del if si el punt pressionat està dintre els marges de pressió. Amb aquesta instrucció s'eviten les falses lectures de pressió.
- **`p.x = map(p.x, TS_MINX, TS_MAXX, 0, 239)`:** Aquesta instrucció mapeja els valors verticals (x) de resistència, pel tamany en píxels de la pantalla, que en aquest cas és de 240 píxels, així a l'hora de dibuixar i pressionar tindrem les mateixes coordenades. En lloc de 239, també podem escriure tftheight(), però en aquest cas no carreguem les llibreries del LCD i és el mateix.
- **`p.y = map(p.y, TS_MINY, TS_MAXY, 0, 319)`:** El mateix que l'anterior instrucció però en aquest cas sobre els valors horitzontals (y) amb un tamany de 320 píxels. En lloc de 320, també es pot escriure tftwidht(), però com s'ha comentat no carreguem les llibreries del LCD. Si ens fixem, els valors horitzontals i els valors verticals, estan intercanviats, aquest aspecte es detallarà al capítol 9.
- **`Serial.print("X = "); Serial.print(p.x)`:** Amb aquesta instrucció, s'envia el valor del punt de pressió obtingut vertical al port sèrie.
- **`Serial.print("\tY = "); Serial.print(p.y)`:** Com l'anterior instrucció però per el punt de pressió horitzontal obtingut.
- **`Serial.print("\tPressure = "); Serial.println(p.z)`:** Com les dues instruccions anteriors però per la quantitat de pressió realitzada.

Un cop vist el codi anterior i les funcions de la llibreria touchscreen; a mode de resum, podem dir que llegim indefinidament, ja que es troba dins la funció loop() del codi, el punt de pressió a la pantalla, si existeix o es troba dins les marges de pressió, es desa a p, es mapejen les coordenades X i Y per a que obtinguem el mateix resultat que el valor de tamany en píxels i els envia al port sèrie per a que es puguin veure. El resultat d'aquest codi executat al Arduino, el podem veure amb el monitor sèrie a la figura 32.

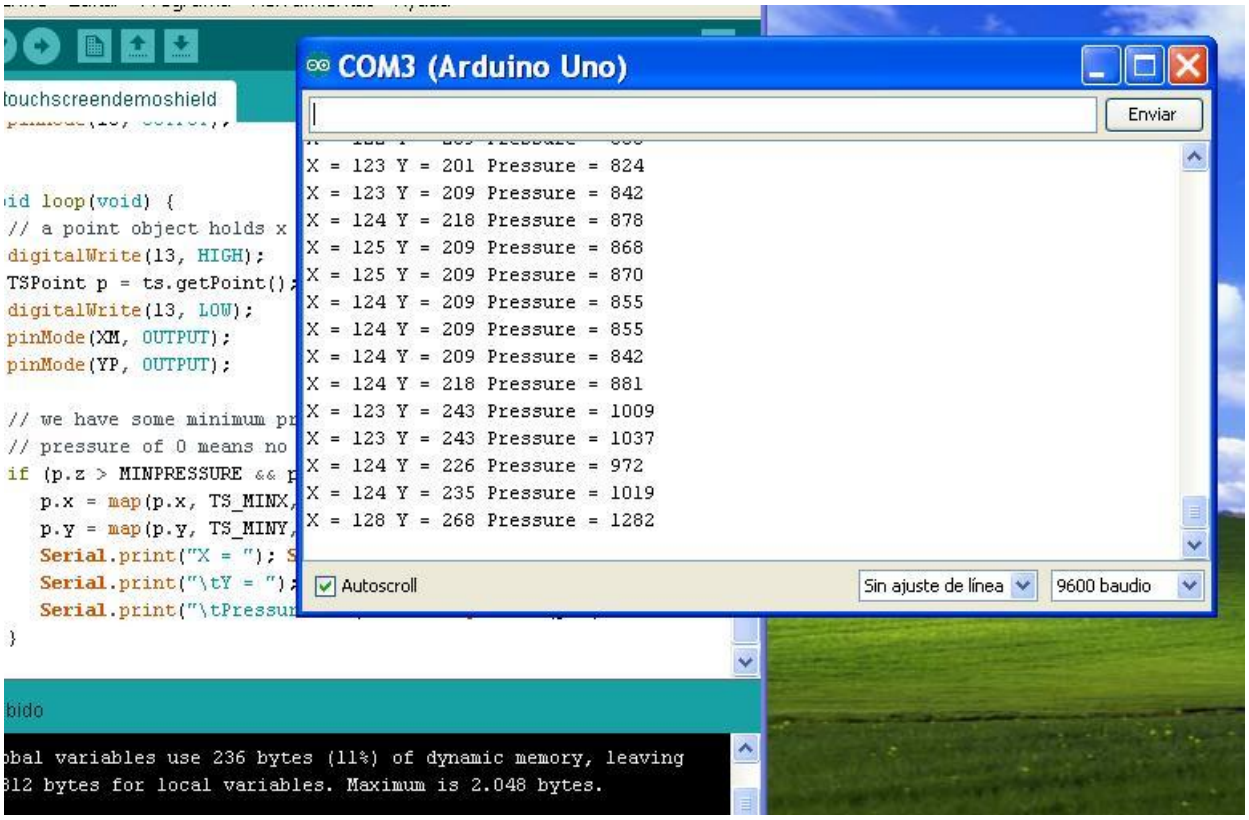


Figura 32: Resultat de l'exemple del codi anterior.

Com es pot observar a la figura, el punt de pressió que s'està exercint, en el moment de la captura, està en la part central de la pantalla aproximadament. Per tant amb les funcions vistes anteriorment i les llibreries instal·lades, podem fer ús de la touchscreen en altres programes més complexos i ja coneixem el seu funcionament.

5.3.Funcions per dibuixar a la pantalla.

Com ara ja es coneixen les característiques de la pantalla i les connexions existents entre el driver i la placa d'Arduino, el següent i últim pas és poder comprendre les funcions de software per dibuixar a la pantalla correctament.

Per poder realitzar dibuixos a la pantalla, primer s'han d'instal·lar les llibreries necessàries que han d'acompanyar al driver, aquestes llibreries són:

- Adafruit GFX library [COMDSP-17].
- Llibreria modificada SWTFT [COMDSP-18].

La segona llibreria també ha estat modificada per mi mateix per tal de que funcioni, s'explica al capítol 9. De la mateixa forma que es va realitzar a l'apartat anterior, s'han de descarregar i descomprimir al directori anomenat library dins de la carpeta d'instal·lació del programa Arduino, d'aquesta manera qualsevol altre programa tindrà accés a aquestes sense haver d'incloure-les físicament.

Com anteriorment, per entendre les funcions e instruccions per dibuixar a la pantalla LCD, ens ajudarem d'un exemple que funciona correctament, aquest exemple s'anomena graphictest i es troba dins la llibreria SWTFT baixada anteriorment, tal com es mostra a la figura 33.

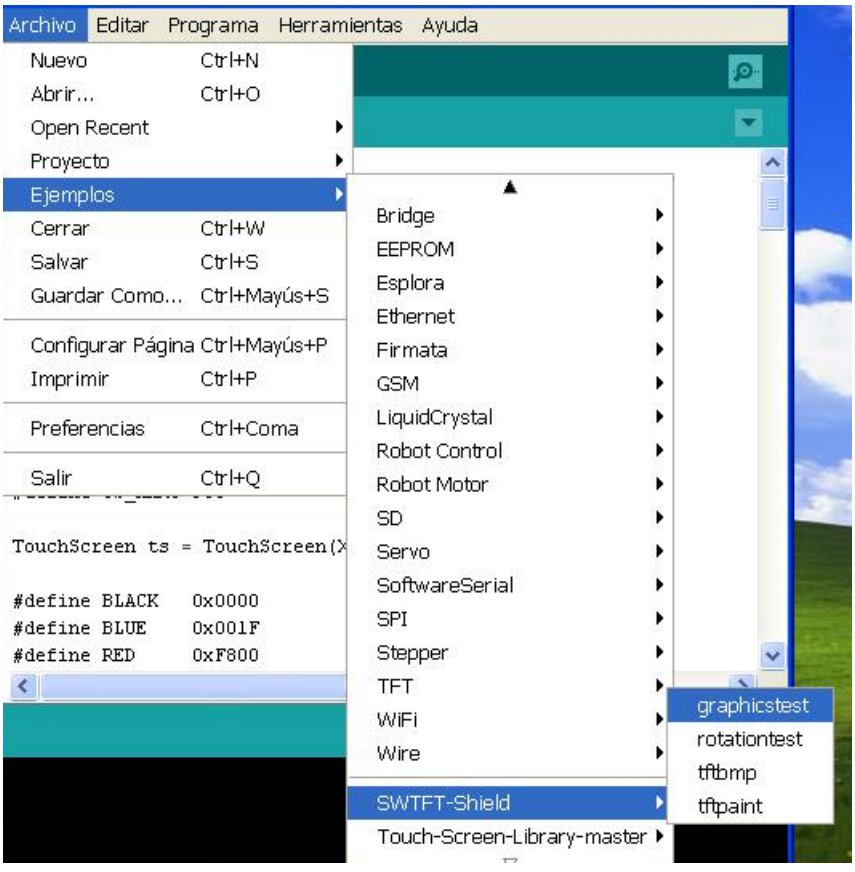


Figura 33: Programa d'exemple funcions LCD-TFT

Amb aquest exemple, veurem totes les funcions disponibles per dibuixar de forma senzilla. Existeix per exemple, una funció de dibuix molt útil, que es tracta de dibuixar un píxel individual, mitjançant un dels tres colors possibles (RGB: Vermell-Verd-Blau) però que no la veurem per dos motius, no la utilitzarem en aquest projecte i te utilitat en dibuixos molt més complexes, com per exemple imatges.

El codi de l'exemple, un cop carregat, és el següent:

```
#include <Adafruit_GFX.h>
#include "SWTFT.h"

#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

SWTFT tft;

void setup(void) {

Serial.begin(9600);
Serial.println(F("TFT LCD test"));
tft.reset();
uint16_t identifier = tft.readID();
   Serial.print(F("LCD driver chip: "));
   Serial.println(identifier);
tft.begin(identifier);
Serial.println(F("Benchmark                Time (microseconds)"));
Serial.print(F("Screen fill              "));
Serial.println(testFillScreen());
delay(500);
Serial.print(F("Text                     "));
Serial.println(testText());
delay(3000);
Serial.print(F("Lines                    "));
Serial.println(testLines(CYAN));
delay(500);
Serial.print(F("Horiz/Vert Lines         "));
Serial.println(testFastLines(RED, BLUE));
delay(500);
Serial.print(F("Rectangles (outline)     "));
Serial.println(testRects(GREEN));
delay(500);
Serial.print(F("Rectangles (filled)      "));
Serial.println(testFilledRects(YELLOW, MAGENTA));
delay(500);
Serial.print(F("Circles (filled)         "));
Serial.println(testFilledCircles(10, MAGENTA));
Serial.print(F("Circles (outline)        "));
```

```
Serial.println(testCircles(10, WHITE));
delay(500);
Serial.print(F("Triangles (outline)      "));
Serial.println(testTriangles());
delay(500);
Serial.print(F("Triangles (filled)       "));
Serial.println(testFilledTriangles());
delay(500);
Serial.print(F("Rounded rects (outline)  "));
Serial.println(testRoundRects());
delay(500);
Serial.print(F("Rounded rects (filled)   "));
Serial.println(testFilledRoundRects());
delay(500);
Serial.println(F("Done!"));

}

void loop(void) {

   for(uint8_t rotation=0; rotation<4; rotation++) {
   tft.setRotation(rotation);
   testText();
   delay(2000);
   }

}

unsigned long testFillScreen() {

   unsigned long start = micros();
   tft.fillScreen(BLACK);
   tft.fillScreen(RED);
   tft.fillScreen(GREEN);
   tft.fillScreen(BLUE);
   tft.fillScreen(BLACK);
  return micros() - start;

}

unsigned long testText() {

   tft.fillScreen(BLACK);
   unsigned long start = micros();
   tft.setCursor(0, 0);
   tft.setTextColor(WHITE);  tft.setTextSize(1);
   tft.println("Hello World!");
   tft.setTextColor(YELLOW); tft.setTextSize(2);
   tft.println(1234.56);
   tft.setTextColor(RED);    tft.setTextSize(3);
   tft.println(0xDEADBEEF, HEX);
   tft.println();
   tft.setTextColor(GREEN);
   tft.setTextSize(5);
   tft.println("Groop");
   tft.setTextSize(2);
```

```
    tft.println("I implore thee,");
    tft.setTextSize(1);
    tft.println("my foonting turlingdromes.");
    tft.println("And hooptiously drangle me");
    tft.println("with crinkly bindlewurdles,");
    tft.println("Or I will rend thee");
    tft.println("in the gobberwarts");
    tft.println("with my blurglecruncheon,");
    tft.println("see if I don't!");
    return micros() - start;

}

unsigned long testLines(uint16_t color) {

unsigned long start, t;
int           x1, y1, x2, y2,
              w = tft.width(),
              h = tft.height();
tft.fillScreen(BLACK);
x1 = y1 = 0;
y2    = h - 1;
start = micros();
for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
x2    = w - 1;
for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
t     = micros() - start; // fillScreen doesn't count against timing
tft.fillScreen(BLACK);
x1    = w - 1;
y1    = 0;
y2    = h - 1;
start = micros();
for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
x2    = 0;
for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
t    += micros() - start;
tft.fillScreen(BLACK);
x1    = 0;
y1    = h - 1;
y2    = 0;
start = micros();
for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
x2    = w - 1;
for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
t    += micros() - start;
tft.fillScreen(BLACK);
x1    = w - 1;
y1    = h - 1;
y2    = 0;
start = micros();
for(x2=0; x2<w; x2+=6) tft.drawLine(x1, y1, x2, y2, color);
x2    = 0;
for(y2=0; y2<h; y2+=6) tft.drawLine(x1, y1, x2, y2, color);
return micros() - start;

}
```

```
unsigned long testFastLines(uint16_t color1, uint16_t color2) {

   unsigned long start;
   int           x, y, w = tft.width(), h = tft.height();
   tft.fillScreen(BLACK);
   start = micros();
   for(y=0; y<h; y+=5) tft.drawFastHLine(0, y, w, color1);
   for(x=0; x<w; x+=5) tft.drawFastVLine(x, 0, h, color2);
   return micros() - start;

}

unsigned long testRects(uint16_t color) {

   unsigned long start;
   int           n, i, i2,
                 cx = tft.width()  / 2,
                 cy = tft.height() / 2;

   tft.fillScreen(BLACK);
   n     = min(tft.width(), tft.height());
   start = micros();
   for(i=2; i<n; i+=6) {
    i2 = i / 2;
    tft.drawRect(cx-i2, cy-i2, i, i, color);
   }

   return micros() - start;

}

unsigned long testFilledRects(uint16_t color1, uint16_t color2) {

   unsigned long start, t = 0;
   int           n, i, i2,
                 cx = tft.width()  / 2 - 1,
                 cy = tft.height() / 2 - 1;
   tft.fillScreen(BLACK);
   n = min(tft.width(), tft.height());
   for(i=n; i>0; i-=6) {
      i2    = i / 2;
      start = micros();
      tft.fillRect(cx-i2, cy-i2, i, i, color1);
      t    += micros() - start;
      // Outlines are not included in timing results
      tft.drawRect(cx-i2, cy-i2, i, i, color2);
   }

   return t;

}

unsigned long testFilledCircles(uint8_t radius, uint16_t color) {

   unsigned long start;
```

```
    int x, y, w = tft.width(), h = tft.height(), r2 = radius * 2;
    tft.fillScreen(BLACK);
    start = micros();
    for(x=radius; x<w; x+=r2) {
        for(y=radius; y<h; y+=r2) {
        tft.fillCircle(x, y, radius, color);
        }
    }

    return micros() - start;

}

unsigned long testCircles(uint8_t radius, uint16_t color) {

    unsigned long start;
    int           x, y, r2 = radius * 2,
                  w = tft.width()  + radius,
                  h = tft.height() + radius;
    // Screen is not cleared for this one -- this is
    // intentional and does not affect the reported time.
    start = micros();
    for(x=0; x<w; x+=r2) {
        for(y=0; y<h; y+=r2) {
        tft.drawCircle(x, y, radius, color);
        }
    }

    return micros() - start;

}

unsigned long testTriangles() {

    unsigned long start;
    int             n, i, cx = tft.width()  / 2 - 1,
                    cy = tft.height() / 2 - 1;
    tft.fillScreen(BLACK);
    n     = min(cx, cy);
    start = micros();
    for(i=0; i<n; i+=5) {
        tft.drawTriangle(
        cx    , cy - i, // peak
        cx - i, cy + i, // bottom left
        cx + i, cy + i, // bottom right
        tft.color565(0, 0, i));
    }

    return micros() - start;

}

unsigned long testFilledTriangles() {

    unsigned long start, t = 0;
    int             i, cx = tft.width()  / 2 - 1,
```

```
                cy = tft.height() / 2 - 1;
    tft.fillScreen(BLACK);
    start = micros();
    for(i=min(cx,cy); i>10; i-=5) {
      start = micros();
      tft.fillTriangle(cx, cy - i, cx - i, cy + i, cx + i, cy + i,
      tft.color565(0, i, i));
      t += micros() - start;
      tft.drawTriangle(cx, cy - i, cx - i, cy + i, cx + i, cy + i,
      tft.color565(i, i, 0));
    }

    return t;

  }

  unsigned long testRoundRects() {

    unsigned long start;
    int           w, i, i2,
                  cx = tft.width()  / 2 - 1,
                  cy = tft.height() / 2 - 1;
    tft.fillScreen(BLACK);
    w     = min(tft.width(), tft.height());
    start = micros();
    for(i=0; i<w; i+=6) {
      i2 = i / 2;
      tft.drawRoundRect(cx-i2, cy-i2, i, i, i/8, tft.color565(i, 0,
0));
    }

    return micros() - start;

  }

  unsigned long testFilledRoundRects() {

    unsigned long start;
    int           i, i2,
                  cx = tft.width()  / 2 - 1,
                  cy = tft.height() / 2 - 1;
    tft.fillScreen(BLACK);
    start = micros();
    for(i=min(tft.width(), tft.height()); i>20; i-=6) {
      i2 = i / 2;
      tft.fillRoundRect(cx-i2, cy-i2, i, i, i/8, tft.color565(0, i,
0));
    }

    return micros() - start;

  }
```

Com es pot apreciar el codi és força extens, per tant, primer s'explicaran les funcions per inicialitzar la pantalla TFT i després les funcions per dibuixar, deixant de

banda la part matemàtica de cada funció. Per tant, les funcions per tal d'inicialitzar la pantalla per poder dibuixar i el resum descriptiu el realitzem a continuació:

- `#include <Adafruit_GFX.h>`: Inclou la llibreria d'Adafruit GFX al nostre programa.
- `#include "SWTFT.h"`: Inclou la llibreria SWTFT al nostre programa.
- `define BLACK 0x0000`: Aquesta instrucció defineix un nom de color, en aquest cas negre (BLACK) amb un codi de 16 bits RGB, per tal de fer-ho més senzill. No explicarem la resta d'instruccions que defineixen els colors per ser idèntiques.
- `SWTFT tft`: Aquesta instrucció serveix per carregar correctament els terminals adients del mòdul LCD amb l'Arduino.
- `tft.reset ()`: Aquesta funció, com el seu nom indica, serveix per esborrar tots els registres del driver del LCD i per tant començar a dibuixar des de zero.
- `uint16_t identifier = tft.readID()`: Aquesta funció serveix per llegir l'identificador de driver que incorpora el LCD, la llibreria Adafruit GFX ens la demana per defecte, com a mesura de seguretat de driver compatible.
- `tft.begin(identifier)`: Aquesta funció envia l'identificador de driver llegit anteriorment a la llibreria per tal d'inicialitzar la pantalla.
- `Delay(x)`: Aquesta funció no te res a veure amb la pantalla, és a dir no és una funció de les llibreries però és important en el context de les pantalles LCD, perquè executa un retard en temps indicat entre parèntesi i en mil·lisegons.

Amb aquestes funcions anteriors, ja tenim inicialitzat el TFT i ja es pot començar a dibuixar. A continuació, es resumeixen les funcions per tal de realitzar-ho:

- `tft.setRotation(rotation)`: Aquesta funció ens permet rotar les coordenades de la pantalla, es realitza col·locant un número en el parèntesi entre 0 i 3, és a dir, realitza una rotació respecte les quatre cantonades de la pantalla o de 90º cada augment de nombre.
- `start = micros()`: Aquesta funció no pertany a les llibreries del LCD, si no al propi llenguatge de l'Arduino i serveix per indicar el temps en microsegons de durada del codi existent entre aquesta i return micros.
- `tft.fillScreen(BLACK)`: Aquesta funció de la llibreria TFT, ens permet dibuixar o col·locar tots els píxels de la pantalla d'un sol color, que és el que es troba entre parèntesi.
- `tft.setCursor(0, 0)`: Aquesta funció situa el cursor del dibuix en les coordenades que es troben entre parèntesis.

- **`tft.setTextColor(WHITE)`:** Aquesta funció estableix el color del text a representar, el color s'indica entre els parèntesis.
- **`tft.setTextSize(1)`:** Aquesta funció estableix el tamany del text, s'indica amb un número del 0 al 9 dins el parèntesi.
- **`tft.println("Hello World!")`:** Aquesta funció escriu el que hi ha entre els parèntesis, pot ésser text, entre cometes, o el valor d'una variable.
- **`tft.width()`:** Aquesta funció retorna el nombre total de píxels horitzontals disponibles a la pantalla, en el nostre cas és de 320.
- **`tft.height()`:** Aquesta funció és idèntica a l'anterior però la orientació és vertical, que en el nostre cas és de 240.
- **`tft.drawLine(x1, y1, x2, y2, color)`:** Aquesta funció ens dibuixa una línia amb les coordenades indicades entre els parèntesis, com es pot entendre, es necessiten dos punts amb dues coordenades cadascun per tal de poder dibuixar-la, i l'últim valor és el color de la línia.
- **`tft.drawFastHLine(0, y, w, color1)`:** Aquesta funció dibuixa el que s'anomena una línea ràpida i es refereix a una línea que es dibuixa de qualsevol forma entre un valor horitzontal x i un valor vertical y, que pot passar per qualsevol altre punt de la pantalla, sempre respectant la longitud especificada i també el seu color, tot això es col·loca entre els parèntesis.
- **`tft.drawRect(cx-i2, cy-i2, i, i, color)`:** Aquesta funció dibuixa un rectangle que necessita de quatre coordenades i un color de contorn, indicades entre els parèntesis. Les dues primeres coordenades pertanyen a dos valors horitzontals i les dues següents al tamany en píxels que serà l'alçada límit de les dues anteriors.
- **`tft.fillRect (cx-12, cy-12, i, i, color1)`:** Aquesta funció realitza la mateixa acció que l'anterior, excepte que el color escollit, és sòlid i dibuixa tot l'espai interior del rectangle.
- **`tft.fillCircle(x, y, radius, color)`:** Aquesta funció ens dibuixa un cercle sòlid, és a dir ens dibuixa tot l'interior del mateix color; només s'ha d'indicar el punt central del cercle mitjançant les coordenades x i y, i el radi. Totes les dades es col·loquen dins el parèntesi.
- **`tft.drawCircle(x, y, radius, color)`:** Aquesta funció realitza la mateixa acció que l'anterior, exceptuant que només ens dibuixa el contorn del cercle.
- **`tft.drawTriangle(cx, cy-i, cx-i, cy+i, cx+i, cy+i, tft.color565(i, i, 0))`:** Aquesta funció ens dibuixa un triangle, els valors que s'han de posar entre parèntesi i són tres punts amb coordenades x i y cadascun i per últim el color del contorn.
- **`tft.fillTriangle(cx, cy-i, cx-i, cy+1, cx+i, cy+i, tft.color565(0,i,i))`:** Aquesta funció realitza la mateixa acció que

l'anterior, però la diferència és que el color és sòlid i ens dibuixa tot l'interior del mateix color.

- `tft.drawRoundRect(cx-12. Cy-12, i, i, i/8, tft.color565(i, 0, 0))`: Aquesta funció ens dibuixa un rectangle de vores arrodonides i ens pinta el contorn del color indicat entre parèntesis. La vora arrodonida s'indica mitjançant un valor petit comparat amb l'alçada.
- `tft.fillRoundRect(cx-i2, cy-i2, i, i, i/8, tft.color565(0, i, 0));`
- `tft.fillRoundRect(cx-12, cy-12, i, i, i/8, tft.color565(0, i, 0))`: Aquesta última funció, realitza la mateixa acció que l'anterior, excepte que ens dibuixa un rectangle de vores arrodonides amb color sòlid.

Ara que ja hem vist les funcions de dibuix disponibles, el resum que es pot fer d'aquest exemple és que, s'inicialitza la pantalla LCD i acte seguit, dins de la funció setup(), pròpia de l'Arduino, anem cridant una a una les funcions que hi han a sota de la funció loop(). Aquestes funcions, ens retornen un nombre de tipus long sense signe (unsignedlong), que pertany al retorn de la funció micros, és a dir el temps que triga en executar cadascuna i com les cridem dins de la funció setup(), doncs només s'executen un cop. Per acabar, dins la funció loop() només s'executa la funció testext() de forma indefinida, canviant la rotació de la pantalla cíclicament.

El resultat, aquest cop només es pot realitzar de forma visual i es mostra a la figura 34, les fotografies presses a la pantalla:

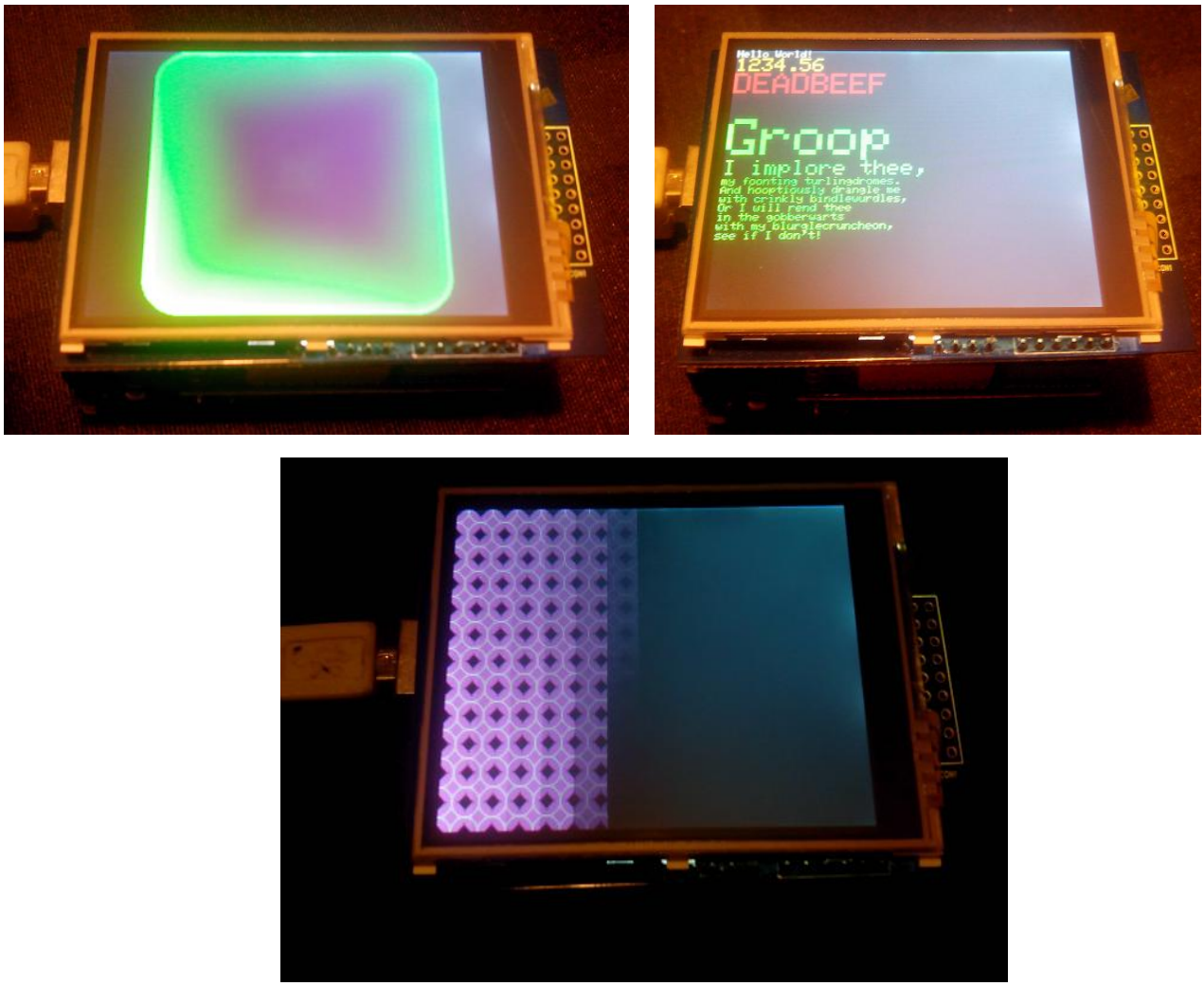


Figura 34: Fotografies del mòdul LCD-TFT executant l'exemple vist.

Un cop vist el funcionament de i programació al complert del mòdul LCD-TFT tàctil, ja podem donar per acabat aquest apartat i capítol, on s'ha consolidat un altre objectiu i és conèixer el funcionament dels mòduls, ara només ens manca l'objectiu final de tot el projecte que es realitzar el programa final i comunicar amb la DSP.

6.Demostració i explicació del programa complert.

A continuació, s'exposa el codi del programa complert aconseguit:

```
//Programa final completat per a PFC

//Per Genís Caminal Villodres

#include <Adafruit_GFX.h>
#include "SWTFT.h"
#include <TouchScreen.h>

#define YP A1
#define XM A2
#define YM 7
#define XP 6

#define TS_MINX 120
#define TS_MINY 870
#define TS_MAXX 850
#define TS_MAXY 908

TouchScreen ts = TouchScreen(XP, YP, XM, YM, 365);

#define BLACK   0x0000
#define BLUE    0x001F
#define RED     0xF800
#define GREEN   0x07E0
#define CYAN    0x07FF
#define MAGENTA 0xF81F
#define YELLOW  0xFFE0
#define WHITE   0xFFFF

SWTFT tft;
char inici='0';
int motor=2, m=2;

void setup(void) {
  Serial.begin(9600);
  tft.reset();
   uint16_t identifier = tft.readID();
  tft.begin(identifier);
  delay(20);
  pinMode (13, OUTPUT);
  upcLogo();
}

#define MINPRESSURE 200
#define MAXPRESSURE 1600
```

```
void loop() {

     if (Serial.available() > 0){
      inici=Serial.read();
      if  (inici == 'r'){
      Serial.print(inici);
      motor=pantElec();
      if (motor == 1) {
        m=Serial.read();
        pantRead();
        }
      }
     }
     else {
     modeOff();
   }
}

void upcLogo(){
   tft.fillScreen(WHITE);
   tft.setCursor(90, 80);
   tft.setTextColor(BLUE);
   tft.setRotation(3);
   tft.setTextSize(9);
   tft.print("UPC");
   delay(2000);
}
int pantElec(){
  tft.fillScreen(BLACK);
  tft.drawRect (40, 80, 80, 80, WHITE);
  int resultat=2;
  int tritochx1=81, tritochx2=163, tritochy1=226, tritochy2=293 ;
  tft.drawRect (200, 80, 80, 80, WHITE);
  int cirtochx1=81, cirtochx2=163, cirtochy1=60, cirtochy2= 160;
  tft.fillCircle (240, 120, 30, RED);
  tft.fillTriangle (110, 120, 50, 90, 50, 150, BLUE);
  for (int i=0 ; i<1000; i++){
     digitalWrite(13, HIGH);
     TSPoint p = ts.getPoint();
     digitalWrite(13, LOW);
     pinMode(XM, OUTPUT);
     pinMode(YP, OUTPUT);
     if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
       p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.height());
       p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.width());
     if ((p.x >= tritochx1) && (p.x <= tritochx2) && (p.y >= tritochy1)
&& (p.y <= tritochy2)) {
         resultat=1;
        delay (200);
     }
     if ((p.x >= cirtochx1) && (p.x <= cirtochx2) && (p.y >= cirtochy1)
&& (p.y <= cirtochy2)){
         resultat=0;
        delay (200);
  }
}
```

```
}
return resultat;
}
void pantRead(){
  int buit[3];
  int velocitat[4];
  int corrent[2];
  int angle[2];
  int fi[5];
  tft.fillScreen(BLACK);
  boolean surt=false;
  int i=0, k=0, a=0;
  while (!surt){
      a=Serial.read();
      if (inici == 'r'){
      Serial.print(inici);
      inici=Serial.read();
      if (inici == '1'){
      Serial.print(motor);
      }
    }
    if (a == '0'){
      for (i=0; i<3; i++){
      buit[i]=Serial.read();
      }
    }
    while (inici != '@'){
    i=0;
    for (i=0; i<4; i++){
      velocitat[i]=Serial.read();
    }
    i=0;
    for (k=0; k<2; k++){
        corrent[k]=Serial.read();
    }
    k=0;
    for (k=0; k<2; k++){
        angle[k]=Serial.read();
    }
     for (int z=0; z<5; z++){
         fi[z]=Serial.read();
    }
    inici= Serial.read();
    Serial.print('@');

    tft.setCursor(0,0);
    tft.setTextSize(2);
    tft.setTextColor(RED);
    tft.print("VELOCITAT: ");
    tft.print (velocitat[0]);
    tft.print (velocitat[1]);
    tft.print (velocitat[2]);
    tft.print (velocitat[3]);
    tft.print (" RPM");
    tft.setCursor(0,80);
    tft.setTextSize(2);
```

```
    tft.setTextColor(YELLOW);
    tft.print("CORRENT:  ");
    tft.print (corrent[0]);
    tft.print (corrent[1]);
    tft.print(" A");
    tft.setCursor(0,0);
    int tochx1= 230,tochx2=80, tochy1= 280,tochy2= 130;
    tft.fillRect(230, 80, 50, 50, WHITE);
     digitalWrite(13, HIGH);
     TSPoint p = ts.getPoint();
     digitalWrite(13, LOW);
     pinMode(XM, OUTPUT);
     pinMode(YP, OUTPUT);
     if (p.z > MINPRESSURE && p.z < MAXPRESSURE) {
       p.x = map(p.x, TS_MINX, TS_MAXX, 0, tft.height());
       p.y = map(p.y, TS_MINY, TS_MAXY, 0, tft.width());
       if ((p.y >= tochx1) && (p.y <= tochx2) && (p.x >= tochy1) && (p.x
<= tochy2)){
           surt= true;
       }
     }
    tft.setCursor(0,160);
    tft.setTextSize(2);
    tft.setTextColor(CYAN);
    tft.print("ANGLE:  ");
    tft.print (angle[0]);
    tft.print (angle[1]);
    tft.print (" GRAUS");
    delay (1500);
    tft.fillScreen(BLACK);
    }

  }
}
void modeOff(){
  tft.fillScreen(BLACK);
  tft.setCursor (90,60);
  tft.setTextColor(RED);
  tft.setTextSize(3);
  tft.println("ATENCIO!");
  tft.println(" ");
  tft.println("    DISPOSITIU");
  tft.println("   NO OPERATIU!");
  delay (500);
}
```

Com el codi es força extens, s'anirà explicant per parts amb fotografies com a demostració. Com es pot apreciar fins a la funció setup(), s'inicialitzen les llibreries i les funcions dels dos mòduls, que ara els connectem al mateix moment, i ja s'ha vist en apartats anteriors. Per tant, la primera funció nova que ens trobem, és upcLogo(), el resultat d'aquesta funció a l'executar-la, es mostra a la figura 35.

Figura 35: Demostració de la funció upcLogo() del programa final.

Com es pot veure a la figura, aquesta funció només serveix per dibuixar el logotip de la universitat durant 2 segons i és la primera funció que s'executa de tot el programa. El codi d'aquesta funció es troba després de la funció loop(), i s'ha dissenyat com a funció externa per entendre'l millor i que el programa principal quedi més polit. És una funció que no ens retorna cap valor, simplement es dibuixa el que es veu a la figura durant dos segons amb les funcions vistes al capítol anterior..

A continuació ens trobem ja dins de la funció loop(), on s'executen tres funcions aïllades, una pantalla de selecció (pantElect()), la funció principal de comunicació (pantRead()) i una funció auxiliar que ens indica que no hi ha comunicació o s'ha acabat la mateixa (modeOff()). Per tant les descriurem per parts i com s'executen dins la funció loop().

Per explicar com s'executen, abans s'ha de tindre clar el protocol dissenyat i la trama de comunicació entre la DSP i l'Arduino. La trama i protocol de comunicació que es realitzarà serà el que es mostra de forma gràfica a la figura 36.

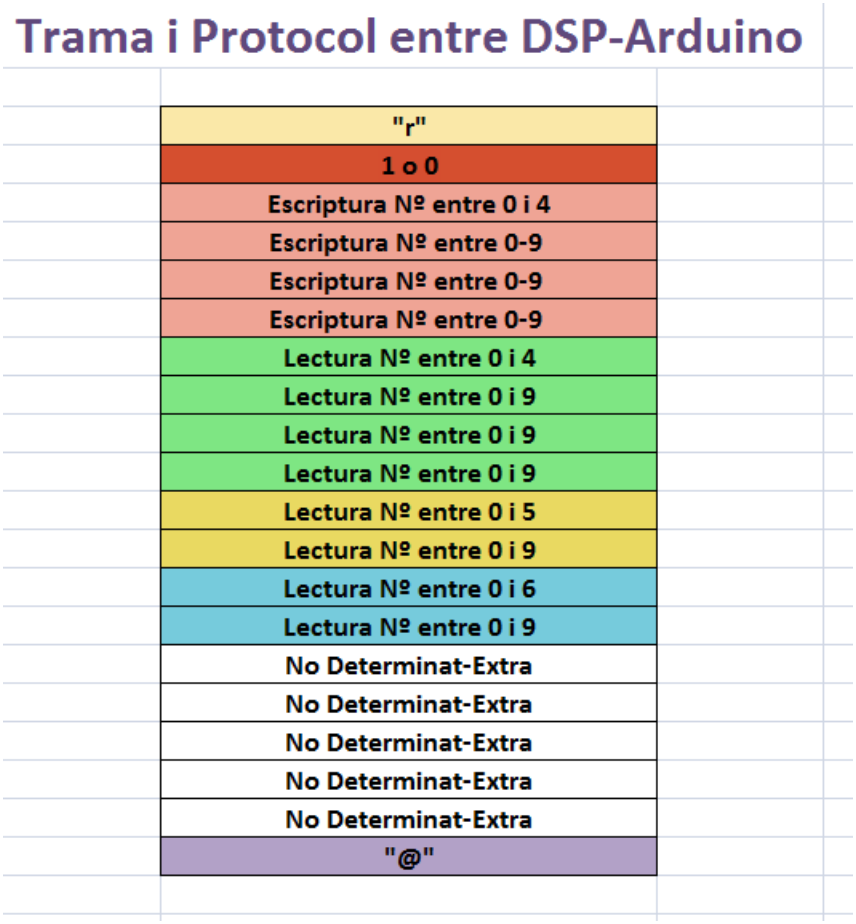


Figura 36: Protocol i trama de comunicació DSP-Arduino.

Per tal d'entendre el protocol dissenyat, s'ha d'observar la figura 36 de dalt a baix. Aquesta figura només representa una trama de comunicació de 20 bytes que es van repetint a mesura que es llegeix o s'escriu, és a dir, quan arribem a la última posició tornem a la primera de forma indefinida. A continuació, s'explica el significat de cada color assignat a un valor determinat:

- Primera posició, en color groc clar i el caràcter "r": aquesta és la primera posició de la trama i per tant ens indica el inici de la comunicació entre els dos dispositius. A més aquest caràcter es confirma per les dues parts, és a dir, s'envia a la DSP des de l'Arduino i a la inversa. En el codi el realitza la variable inici, que llegeix el que li arriba de la DSP, si li arriba el caràcter 'r', li retorna per inicialitzar la comunicació.
- La segona posició, en color vermell i els números 1 o 0: es refereix a la variable que activa el motor, un 1 enviat per l'Arduino encendrà el motor i al contrari amb el 0. En el codi anterior, per tal d'encendre o apagar el motor, hem d'accedir a la pantalla de selecció pantElec().
- Les quatre posicions següents de color vermell clar amb dígits: es refereix a 4 caràcters que seran enviats per l'Arduino per tal de modificar les

revolucions o velocitat del motor, en un començament, es troben inicialitzats a zero dins la funció pantRead(), ja que no ha donat temps a escriure aquesta part del programa, s'explica al capítol 8. El rang, es representa de major a menor alçada en la figura i estarà comprès entre 0 i 4095.

- Les altres quatre posicions següents de color verd amb dígits: es refereix també a la velocitat, però aquest cop de lectura, és a dir, la DSP enviarà aquests valors de revolucions a l'Arduino per a que els mostri per pantalla; i te el mateix rang que l'anterior, és a dir, entre 0 i 4095 revolucions per minut, també es representa de major a menor alçada. Al codi, es troba també dins la funció pantRead().
- Les dues següents posicions de color groc: són també de lectura i es refereix al valor de la corrent del motor en ampers, te un rang determinat de 0 a 50 (A), per tant l'Arduino rebrà les dades des de la DSP i les representarà en pantalla. El codi també es troba dins la funció pantRead().
- Les següents dues posicions de color blau: es refereixen a l'angle de conducció dels transistor de l'inversor i tenen un rang de 0 a 60º. També són valors de lectura i es representaran a la pantalla LCD i per tant, el codi es troba dins la funció pantRead().
- Les cinc següents posicions de color blanc: No tenen cap tipus de referència, només són 5 bytes a lliure disposició o extres per si en un futur cal afegir variables a visualitzar o escriure, a mode d'ampliació del programa.
- L'última posició de color morat amb el caràcter "@": es refereix al final de la trama en qüestió, i per tant el codi torna a començar des del principi a una altra trama nova, i passa quan es rep aquest caràcter, a més es confirma per les dues bandes. També es troba a la funció pantRead().

Ara que ja es coneix el funcionament de la trama i el protocol, podem continuar descrivint el codi anterior. Una vegada rebut i enviat el caràcter 'r', el codi ens envia a la pantalla de selecció per tal d'escollir si llegirem o escriurem a la DSP, per això la variable motor està assignada a la funció pantElec(), que aquesta ens retorna un valor enter depenent de la nostra elecció. El codi de la funció pantElec(), està dissenyat per tal que l'usuari realitzi l'elecció polsant dues icones dibuixades a la pantalla i mitjançant el panell tàctil, el resultat es mostra a la figura 37.

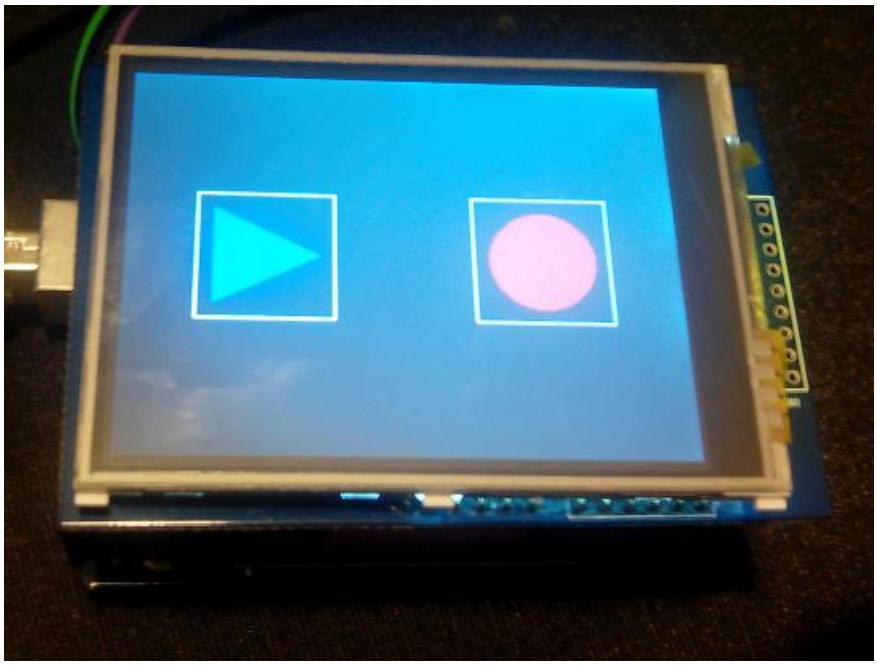


Figura 37: Demostració de la pantalla de selecció.

Com es pot observar a la figura, si l'usuari escull polsar el triangle blau, en el codi, el valor de la variable motor es posarà a 1 mitjançant la detecció de la pulsació de la pantalla tàctil i el codi de la funció TSpoint adientment optimitzat a les coordenades del triangle, preparà a l'Arduino per llegir dades portant aquest a la funció pantRead(). La DSP encendrà el motor quan rebi aquest 1. Si l'usuari escull polsar el cercle vermell, el motor no s'engegarà i l'Arduino es prepararà per escriure el valor de la velocitat a la DSP, però malauradament, la funció pantEsccriu() no està enllestida (explicació al capítol 8).

Un cop l'usuari ha escollit el triangle blau,el codi encén el motor com ja hem comentat i ens porta dins la funció pantRead() on l'Arduino comença a llegir la trama vista i mitjançant els arrays buit, velocitat, corrent i angle, s'anirà omplint d'informació i fins que l'usuari aturi el procés mitjançant un botó quadrat blanc o s'aturi la comunicació, l'Arduino no deixarà de llegir indefinidament, inicialitzant els arrays cada cop que rep el caràcter final de trama, per tant la funció pantRead(), es buida i no ens retorna cap valor, simplement actualitza i dibuixa els arrays. El resultat de llegir dades es mostra a la figura 38.

Figura 38: Demostració de lectura d'una trama

Quan l'Arduino deixi de llegir per qualsevol dels dos motius comentats, es tornarà a la pantalla de selecció inicial.

Per acabar la descripció del codi, ens manca descriure la funció modeOff(), aquesta s'activa quan no existeix comunicació entre la DSP i l'Arduino, pel motiu que sigui, desconnexió, finalització de les trames, etc. Si ens fixem en el codi, aquesta funció s'activa quan el condicional if principal, ubicat a la funció loop() i la instrucció Serial.available deixen d'ésser certes. Aleshores aquesta funció ens escriu a la pantalla cíclicament e indefinidament, mentre no torni la comunicació, un text, tal com es pot apreciar a la figura 39.

Figura 39: Demostració de la funció modeOff().

Fins aquí s'ha descrit el codi de forma molt resumida per no allargar i complicar més el capítol, a més la resta de funcions ja es coneixen de capítols anteriors, on han estat descrites; un aspecte important a tindre en compte, de forma general en el codi, es que s'ha de tindre cura amb el tractament de les variables, ja que depenent de que volem llegir i que s'envia podem cometre errors molt fàcilment, per exemple l'Arduino per defecte envia les dades en format ASCII, però si llegim aquestes com enters i rebem un caràcter, la comunicació es perd, de la mateixa forma que un array amb posicions indeterminades.

Totes les figures anteriors han estat executades des del codi anterior mitjançant MatLab com a DSP virtual, manquen per tant realitzar més proves amb la DSP física. EL codi de MatLab que ens permet llegir una trama, es mostra a continuació:

```
PS
PS.ReadAsyncMode='continuous';
w=[2,0,0,0,0,1,5,0,0,2,0,4,5,0,0,0,0,0];
z='a';
a='a';
fwrite(PS,'r');
z=fscanf(PS,'%c',1);
for i=1:18
fwrite (PS,w(i));
end
fwrite(PS,'@');
a=fscanf(PS,'%c',1);
```

7.Problemàtica i solucions aplicades

Han sorgit varies problemàtiques durant el trajecte d'aquest projecte. En primer lloc l'elecció del mòdul LCD no va ésser molt bona, la decisió final d'escollir un mòdul xinès enlloc de l'equivalent de l'empresa Adafruit, va ésser únicament el cost, que es multiplicava per dos, i la resta de pantalles o eren caríssimes o s'havia d'emprar una plataforma diferent que també augmentava el cost i s'havia d'incloure temps de disseny de hardware i plaques de circuit imprès.

Només arribar a les meves mans el mòdul TFT e intentar provar-lo, va ésser una tasca gairebé impossible, les llibreries donades pel fabricant mai van funcionar (suposo que es van copiar la placa de l'empresa Adafruit i per això no es troba ja a la seva pàgina web) i les que es trobaven a internet tampoc. Per tant amb un suport del fabricant inexistent, zero informació per enlloc i just abans de comprar-ne un altre vaig realitzar un últim intent i va funcionar de la següent manera: resulta que segons tots els datasheets de drivers de pantalles, tenen una adreça de memòria o registre que al llegir-la et retornen la identificació del driver, en el cas d'aquest LCD no va funcionar i retornava valors estranys sense sentit, d'aquesta manera vaig arribar a la conclusió de que els custom chip on estava ubicat el driver desconegut, d'alguna manera o estaven connectats incorrectament o no deixaven accedir a aquesta informació (avui encara ho desconec), de totes formes aquests dos aspectes donaven igual ja que vaig decidir provar els drivers que no acceptaven les llibreries d'Adafruit i vaig trobar una que feia alguna cosa a la pantalla [COMDSP-15]. Descartant, vaig arribar a la conclusió de que el driver inclòs era el ST7781, però tampoc acabava de funcionar i finalment la solució va ésser modificar el firmware del driver directament. El que vaig trobar llegint el datasheet era l'orientació del LCD, on existeixen diferents formes de representar les dades, de dalt a baix, a l'inversa, meitat de la pantalla una orientació i l'altre una diferent, etc., i modificant el firmware vaig trobar una orientació adient i tot va tindre sentit i va funcionar correctament com es pot veure al llarg del projecte. A la pàgina 41 del datasheet es troba aquesta informació, tal com mostra la figura 40.

10.1.3 Device Output Control (R01H)

Device Output Control (R01H)																	
RS	/WR	/RD	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1
1	↑	1	0	0	0	0	0	SM	0	SS	0	0	0	0	0	0	0
Default value			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Description

SS: Select the shift direction of outputs from the source driver.
When SS = 0, the shift direction of outputs is from S1 to S720
When SS = 1, the shift direction of outputs is from S720 to S1.
In addition to the shift direction, the settings for both SS and BGR bits are required to change the Assignment of R, G, B dots to the source driver pins.
To assign R, G, B dots to the source driver pins from S1 to S720, set SS = 0.
To assign R, G, B dots to the source driver pins from S720 to S1, set SS = 1.

Note: When changing SS or BGR bits, DRAM data must be rewritten.

SM: Sets the gate driver pin arrangement in combination with the GS bit (R60h) to select the optimal scan mode for the module.

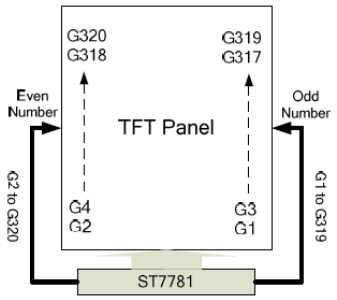
SM	GS	Scan Direction	Gate Output Sequence
0	0		G1,G2,G3,G4…..,G316 G317,G318,G319, G320

Figura 40: Orientació ST7781.

Tal com mostra la figura s'ha de modificar el registre 01h col·locant dos zeros, un en SM i l'altre a SS, per tal de tindre l'orientació correcte. Aquesta modificació es realitza al firmware de la llibreria SWTFT.cpp, tal com es mostra a la figura 41.

```
79
80  static const uint16_t ST7781_regValues[] PROGMEM = {
81     0x0001,0x0000,
82     0x0002,0x0700,
83    0x0003,0x1030,
84    0x0008,0x0302,
85    0x0009,0x0000,
...
123   //-------DISPLAY ON------//
124   0x0007,0x0133,     0x0001,0x0000,
125    0x0002,0x0700,
126   0x0003,0x1030,
127   0x0008,0x0302,
128   0x0009,0x0000,
```

Figura 41: Ubicació del registre d'orientació modificat correctament.

Aquest problema em va tindre dos mesos sense avançar i quan estava a punt de deixar-ho per impossible i adquirir un altre pantalla ho vaig aconseguir.

Un altre problema gros va ésser la touchscreen, resulta que he descobert recentment que la resistència de la coordenada X (vertical a la touchscreen) no és gaire lineal per la part baixa on es troba el rang de (0,0) a (0,240), això feia que no fos gaire responsiva, el programa final ja s'ha actualitzat amb les noves coordenades.

Finalment, a l'hora de realitzar proves de comunicació entre l'Arduino i el MatLab i també amb la DSP, si els arrays i el buffer d'entrada i de sortida, no estaven ben fixats, la comunicació es perdia i era impossible rebre o enviar dades coherents. Per tant, ara es comptabilitza la quantitat de dades que s'envien i es reben i també el tamany de buffer necessàri.

8.Conclusions

Com a conclusions finals que crec importants, hi ha les millores que es podrien implementar per tal d'apropar-se a les tendències de futur que es van comentar anteriorment i l'aprenentatge consolidat i aspectes personals al respecte.

- Millores de futur: segons el meu parer, hi han tres possibles millores a implementar: Miniaturització, comunicació Wireless i Software de la plataforma. La primera, em refereixo a miniaturització, a que és possible reduir la placa de l'Arduino UNO i del mòdul de comunicació a una mínima expressió, dins la mateixa placa i per tant el dispositiu passaria a ésser molt més portàtil. Respecte la comunicació Wireless, un cop realitzada la comunicació amb RS-232, és molt senzill passar a un altre tipus de comunicació canviant el mòdul RS-232 pel medi buscat, sense haver de canviar el protocol o gairebé res, per exemple, per comunicar via freqüència wifi, existeixen uns mòduls a la plataforma Arduino anomenats Xbee que serveixen per tal efecte, també per bluetooth. I per últim, respecte millores de software, es veu clar al finalitzar aquest projecte que manca una millor optimització del mateix i segurament un, també millor, entorn gràfic, encara que no fossin els objectius principals del projecte, per exemple al tindre una pantalla gràfica, es poden representar gràfiques de rendiment de motor amb diferents dades separades per colors. Amb aquestes petites millores, podríem tindre un dispositiu portàtil amb comunicació sense fils i la possibilitat de que un operari amb un únic panell pogués visualitzar i realitzar un mínim control sobre diferents màquines i per tant acostar-nos a un nivell més elevat d'automatització industrial.

- Aprenentatge i aspectes personals: des de que vaig veure el projecte, em va agradar la idea, degut a que el tema de la visualització de dades mitjançant una pantalla LCD, sempre m'ha agradat i durant la carrera aquest tipus de dispositiu no s'arriba a veure. Per tant, ara conec com moure gràfics en aquest tipus de dispositius i tinc un coneixement més profund d'aquestes noves tecnologies. També, el tema de la comunicació amb la DSP, pensava que seria molt més senzill i he aprés com moure informació i senyals de forma més complex. En resum, tinc un coneixement molt més gran del que ja coneixia de la plataforma Arduino i ara puc realitzar programes més complexes, també la gestió d'escollir dispositius nous, on no crec que em torni a passar l'elecció d'un mòdul TFT sense tindre un mínim de suport (per exemple, si fos un projecte d'empresa, el temps dedicat no compensa la diferència de cost d'un mòdul a altre amb bon suport), desgraciadament

m’hagués agradat deixar el projecte més enllestit amb funcions i tècniques que s’han quedat a l’aire. També i com autocrítica, hauria d’haver dedicat més temps a practicar amb la DSP, però el fet d’haver format família i treballar a l’hora amb els horaris actuals, deixen poc marge per la creativitat i el temps necessaris per consolidar un projecte com aquest, que personalment crec que és força ambiciós i aquest detall el fa força atractiu. Per acabar, agrair la paciència del docent Balduí Blanqué i el proper doctorat Marc Gomila, de l’UPC de Vilanova i la Geltrú.

9.Bibliografía

-Capítol Introducció:

[BALD-1]: PDF control SRM con DSC (cortesia de Balduí Blanqué i Marc Gomila)

-Capítol Estat actual dels inversors:

[COMDSP-1]: Web proveïdor Inversors a la venda: http://www.inverter.co.uk/inverters.htm (última visita setembre 2015).

[COMDSP-2]: Manual d'usuari MX2 de OMRON: http://www.proenergo.ru/doc_pdf/drives-doc/mx2/MX2+UsersManual.pdf (última visita setembre 2015).

[COMDSP-3]: Manual d'usuari MicroMaster 440 de SIEMENS: https://www.inverterdrive.com/file/siemens-micromaster-440-manual (última visita setembre 2015).

[COMDSP-4]: Manual d'usuari AC30 de PARKER: http://www.parker.com/literature/SSD%20Drives%20Division%20North%20America/Catalogs%20and%20Brochures/AC30V_Catalog_HA473413_NA.pdf (última visita setembre 2015).

[COMDSP-5]: Link pàgina empresa automatització amb Arduino: http://www.opiron.com/ (última visita setembre 2015).

[COMDSP-6]: Link article "El futuro de la producción" de interempresas.net: http://www.interempresas.net/Robotica/Articulos/134751-El-futuro-de-la-produccion.html (última visita setembre 2015).

-Capítol Descripció de la plataforma:

[COMDSP-7]: Esquema de la placa Arduino UNO: https://www.arduino.cc/en/uploads/Main/arduino-uno-schematic.pdf (última visita setembre 2015)

[COMDSP-8]: Datasheet microcontrolador de l'Arduino UNO: http://www.atmel.com/Images/doc8161.pdf (Última visita setembre 2015)

[COMDSP-9] Pàgina de descarrega oficial del software de programació de l'Arduino: https://www.arduino.cc/en/Main/Software
(Última visita setembre 2015).

[COMDSP-10] Datsheet de l'integrat de l'empresa Maxim MAX3232: http://datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf
(Última visita setembre 2015)

[COMDSP-11] Pàgina oficial del mòdul RS-232 de l'empresa DF ROBOT (datasheets i més informació): http://www.dfrobot.com/index.php?route=product/product&product_id=1030#.VfVausvNuic (Última visita setembre 2015).

[COMDSP-12] Pàgina oficial del fabricant xinés mcufriend del mòdul TFT tàctil: http://www.mcufriend.com/ (Última visita setembre 2015).

[COMDSP-13] Datasheet del driver Sitronix del mòdul LCD TFT de 2,8": http://www.rockbox.org/wiki/pub/Main/SansaFuzePlus/ST7781.pdf
(Última visita setembre 2015).

[COMDSP-14] Link oficial de la guia de programació per la DSP sobre la comunicació sèrie: http://www.ti.com/lit/ug/sprufz5a/sprufz5a.pdf
(Última visita setembre 2015).

-Capítol descripció del funcionament i programació del mòdul LCD-TFT tàctil:

[COMDSP-15] Taula extreta de la pàgina web:
http://www.smokeandwires.co.nz/blog/a-2-4-tft-touchscreen-shield-for-arduino/
(Última visita setembre 2015).

[COMDSP-16] Pàgina web de descàrrega de la llibreria de la touchscreen: https://github.com/adafruit/Touch-Screen-Library (Última visita setembre 2015).

[COMDSP-17] Pàgina web de descàrrega de la llibreria de gràfics pel LCD: https://github.com/adafruit/Adafruit-GFX-Library (Última visita setembre 2015).

[COMDSP-18] Pàgina web de descàrrega de la llibreria de gràfics pel LCD (pròpia d'Adafruit modificada per l'administrador de la pàgina i també per mi mateix, s'explica al capítol 9): https://github.com/Smoke-And-Wires/TFT-Shield-Example-Code (Última visita setembre).